TRS-80 SYSTEM 80 VIDEO GENIE PMC-80

Issue 23, ~~September~~ October 1981

SCOOP: FIRST PICTURE OF GENIE III
(System-80 III?)

Also in this issue:

HARDWARE:
Add Joysticks and Input/Output Ports to your '80-Part 3

PROGRAMMING:
Better Basic Programming — Part 4

SOFTWARE:
PUT-New Command for Level II   Shorten Disk programs
Keyboard Shiftlock   Morse Decoder
Poker

MICRO-80

MICRO-80 is an international magazine devoted entirely to the Tandy TRS-80 microcomputer and the Dick Smith System 80/Video Genie. It is available at the following prices (all prices shown in Aus.$ except for U.K. prices which are in pounds Sterling).

12 months subscription	Aus.	$24.00
	NZ.	$36.00 (Airmail)
	Hong Kong	$46.00 (Airmail)
	U.K.	£16.00
Single Copy	Aus.	$2.50
	N.Z.	$3.50 (Airmail)
	Hong Kong	$4.25 (Airmail)
	U.K.	£1.50
Months programs on cassette	Aus.	$3.50
	N.Z.	$4.00 (Airmail)
	Hong Kong	$4.50 (Airmail)
(at present available from Australia only)	U.K.	$4.75 (Airmail)
12 months subscription to magazine and cassette	Aus.	$60.00
	N.Z.	$78.00 (Airmail)
	Hong Kong	$88.00 (Airmail)
	U.K.	£41.00 (Airmail)

Special bulk purchase rates are also available to computer shops etc. Please use the form in this issue to order your copy or subscription.

The purpose of MICRO-80 is to publish software and other information to help you get the most from your TRS-80, System 80 or Video Genie and their peripherals. MICRO-80 is in no way connected with either the Tandy or Dick Smith organisations.

** WE WILL PAY YOU TO PUBLISH YOUR PROGRAMS **
Most of the information we publish is provided by our readers, to whom we pay royalties. An application form containing full details of how you can use your TRS-80 or System 80 to earn some extra income is included in every issue.

** CONTENT **
Each month we publish at least one applications program in Level I BASIC, one in Level II BASIC and one in DISK BASIC (or disk compatible Level II). We also publish Utility programs in Level II BASIC and Machine Language. At least every second issue has an article on hardware modifications or a constructional article for a useful peripheral. In addition, we run articles on programming techniques both in Assembly Language and BASIC and we print letters to the Editor and new product reviews.

** COPYRIGHT **
All the material published in this magazine is under copyright. That means that you must not copy it, except for your own use. This applies to photocopying the magazine itself or making copies of programs on tape or disk.

** LIABILITY **
The programs and other articles in MICRO-80 are published in good faith and we do our utmost to ensure that they function as described. However, no liability can be accepted for the failure of any program or other article to function satisfactorily or for any consequential damages arising from their use for any purpose whatsoever.

***** CONTENTS *****

***** EDITORIAL *****

This month, we have handed over much of our editorial space to Tony Edwards, our U.K. correspondent. Tony has presented us with a scoop; details and a photograph (our front cover) of the new (Video) Genie III. We will now hand over to Tony then return with some of our own comments later.

***** PEEKing (UK)   by our U.K. Correspondent Tony Edwards *****

Last month I gave a run down of the hardware situation in this country and this month I am continuing in the same vein, but talking about software. The '80 user in this country is very well served with regards ready made software for the '80. There is, of course, the large range of Tandy products, most of which are compatible with non-Tandy machines. My local friendly Tandy store manager does not think that these programs will run in Video Genie machines, however. He told me that the chips are different and 'built to different standards'. I do not believe him, as I have never had any difficulty using Tandy programs on my Video Genie. The only problems are the different keys and sound output.

As well as the Tandy software, there are a number of Software Houses in the U.K. some of which specialize in '80 software. These include Kansas City Systems, Microcomputer Applications, and Molimerx Ltd. All these provide an extensive range of programs from business to games and add new programs regularly. Molimerx (the name is Latin for 'Software') produces an almost club atmosphere in their monthly listings which, as well as over 400 programs, contains sections on problems and useful hints. The hallmark of all these suppliers is the back-up service. All accept the fact that bad tapes will sometimes be sent out and will exchange faulty tapes by return post.

American magazines are widely available here and they provide another source of software. The American suppliers can sell at much lower prices than the U.K. suppliers but, in some cases, the service is not so good and it is difficult to press a complaint over great distances.

Another source which is available is in the small ads. part of the various computing magazines. In these ads., freelance programmers sell their work at very reasonable prices. Some of these programs are very good indeed, and well worth the small cost.

The Genie II (EG 3008) is now available in this country. This machine is very similar to the Video Genie (EG 3033), known in Australia as the System 80. It uses Microsoft Level II BASIC and, as such, is compatible with most of the programs which have been produced for the '80 series of machines. The only obvious incompatibility is the loss of the second cassette. On this new machine the internal cassette has been replaced by a numeric keypad and only one external cassette port is available, thus programs which require two cassettes will not work on the new machine. The standard ROM has been expanded to 13K and now includes additional professional features, terminal features, and special function keys. The keyboard has 71 keys including a full typewriter section, numeric keypad, and four special function keys. Lower case is standard. This looks a fine machine aimed at the business market but I can see many hobby users enjoying it.

No sooner had I finished considering the Genie II than I was invited to see the press release of the Genie III. This machine will not be generally available for twelve months or more but already I think it is a winner. It is intended to be compatible with the Video Genie and the Genie II so I think the suppliers should be congratulated on their attempts to keep all the machines in the stable working together. The Genie III has no ROM being disk based with operating disks carrying the function usually found in ROM. The machine is the first to support both CPM (80 x 24 screen format) and Level II BASIC (64 x 16), thus giving it a huge software access. The machine has 64K of RAM and 1 megabyte of on-line disk, which can be doubled externally if required. It uses a Z80A processor operating at 4MHz and has both parallel and serial interfaces as standard. All this at a projected price of £1600 in the U.K. cannot be bad.

There are a number of new 'stick on' goodies becoming available at this time in the U.K. which I will be covering in a future piece. Lowe Electronics (the suppliers of the Video Genie) are marketing a converter for the TRS-80 to allow it to use Genie add-ons.

- 0000000000 -

On receipt of this story, we immediately rang Jim Rowe, Technical Director of Dick Smith Electronics. We knew Jim had very recently been to the U.K. and it seems that one of the major reasons for that trip was to attend the preview of the Genie III. Jim was able to expand somewhat on the information supplied to Tony.

The unit presented and photographed, is the engineering prototype. If the dealers have their way, the ultimate production models will be rather more compact and visually attractive. Price too, could be rather closer to what it would cost for a similarly optioned Genie II. It is not certain yet whether the BASIC interpreter will be on disk or will be contained in a ROM as at present. No decision has yet been made on the DOS to be used with the system (i.e. when not in CP/M mode). The disk drives operate in double density. Those in the prototype unit were TEAC but production units could have different drives.

What about our own opinion? We think it is a very intelligent move. For some years now, CP/M has been the de facto industry standard disk operating system. A great deal of software has been developed to operate under CP/M, particularly business software, interpreters (BASIC, COBOL, FORTRAN) and utilities. Two relatively recent events have virtually assured CP/M and CP/M compatible systems of a firm place in microcomputing for at least the next five years. First, there was the Z80 Softcard produced by Microsoft for the Apple. This makes Apple computers capable of running CP/M and has lead to the development of a great deal of new software, both business and hobbyist, running under CP/M. Secondly, there is the emerging Japanese microcomputer industry. The greatest weakness the Japanese have in selling computers into Western countries is the availability of suitable software. Almost to a man, Japanese companies have designed their computers to operate under CP/M thus overcoming the problem. Now, programmers all over the world will be able to write programs to operate under CP/M and know that, with only minor modifications, they should run on over 60% of the disk-based microcomputers which will be sold in the next few years. The Genie III will fit right into this pattern and should do well. It will also offer the advantage of allowing the user access to the very large library of TRS-80 Model I software which is available. We think that the Genie III is well conceived and look forward to its early introduction. Incidentally, you may not have to wait 12 months to see them in the shops; apparently the manufacturers are now talking about delivery in April or May 1982.

We had so much to squeeze into the magazine this month that we have had to hold over one of our advertised programs - Towers of Hanoi, until next issue. Part 2 of The Theory and Techniques of Sorting will also appear next month.

-0000000000-

***** BETTER BASIC PROGRAMMING - PART 6   by Rod Stevenson *****

** MORE ASSEMBLY/BASIC **

```
0 '

1 'BREAK=SHIFT B. SHIFT@=SHIFT A. USE SHIFT E FOR ENTER WHEN FOL
LOWS SAVE, CSAVE, LIST, LLIST; OTHERWISE THEY ARE DISABLED. CAN
BE MERGED ONTO FRONT OF ANY BASIC. BEFORE (C)SAVEING, POKE 255 I
NTO 17131-2.
2 SM=PEEK(16549):SL=PEEK(16548):SA=SM*256+SL+8:SL=SAAND255:SM=IN
T(SA/256):FORA=SATOSA+97:READD:POKEA,D:NEXTA
3 T1=PEEK(16405):T2=PEEK(16435):POKE16405,0:POKE16435,201:POKE16
406,SL:POKE16407,SM
4 DATA205,227,3,254,101,40,65,254,98,40,17,254,97,40,16,254,13,4
0,15,254,1,40,3,254,96,192,175,201,62,1,201,62,96,201,217,237,91
,164,64,175,103,46,83,25,235,42,167,64,43,215,26,19,183,250,204,
6,190,35,40,246,26,19,254,128,32,250,26,254,255
5 DATA32,230,217,62,13,201,76,73,83,84,128,76,76,73,83,84,128,67
,83,65,86,69,128,83,65,86,69,128,255
6 POKE16405,T1:POKE16435,T2:POKE17131,0:POKE17132,0:'FOLLOWING I
S A SAMPLE PROGRAM.
7 PRINT" TRY TO STOP!!!";:PRINT@512,;:PRINT"CONNECT AMPLIFIER TO
 AUX PLUG. OR PLUG SPEAKER INTO EAR-SOCKET   & PRESS RECORD-DISA
BLE LEVER ON RECORDER, THEN RECORD & PLAY.
               THEN PRESS ANY KEY.
8 IF INKEY$=""THENGOTO8
9 CLS:FORI=1TO23:READD:S$=S$+CHR$(D):NEXT
10 X=VARPTR(S$):POKE16526,PEEK(X+1):POKE16527,PEEK(X+2)
11 PRINTCHR$(23);"LISTEN ";:X=USR(RND(50))
12 PRINT"LISTEN ";:GOTO11
13 DATA205,127,10,14,140,69,62,1,211,255,16,254,69,62,2,211,255,
16,254,13,32,239,201
```

This program routine is here to illustrate many of the points left over from the last article. Some of these perhaps are outside the scope of the "elementary" nature of this series, but I'll present them anyway! So if you find some a little complex now, leave them till later and you will be glad to be able to understand, at a later date, with more computing "under your belt". While on that subject, I must stress again that true understanding (of this series, or of any other instruction-type book or magazine) really does require "hands on" experience and practice. So, don't just look at the listings in these articles. Type them in, RUN them and try to understand how they work, even before you read my explanation.

WHERE TO KEEP MACHINE LANGUAGE ROUTINES.

As I write machine language only as routines to be accessed from BASIC, this is an important consideration. Of course, if your machine language is stand-alone it won't be - you can just enter it with SYSTEM and the entry point.

There are basically three ways I store machine language:

1. In a definite memory location. Though this can also be a location determined by the program: last episode the program looked at the current top of memory to determine where to put the machine language program. But generally, you will have to protect memory, load in your program (whether by SYSTEM or POKE from BASIC) and it's done. The disadvantage here is that you are relying on the user to remember to set memory size (and realise, the documentation you carefully provided may go astray from the program), and at the correct value. You will also have to provide different programs for all memory configuration. But without a doubt it's the simplest way.

    As well as locating your machine language program at the top of memory, you can put it at the start of BASIC, and relocate the start of BASIC pointers (put the new location into 40A4-5H and NEW to set other pointers). But again, you will have to provide separate programs for disk systems (and different ones for each disk system).

2. In a string. This is actually quite simple, and is satisfactory unless your routine is not relocatable or is more than 250 bytes long. And if it is you won't have been able to use relative jumps (+129 or -126) to make it relocatable so you will need an absolute location known at the time of assembling your program. Yes, I know you could use array space or storage, but this gets quite "hairy" because arrays are not stored contiguously in memory.

    Having decided to use a string for storage, write a DATA line containing your machine language code in decimal numbers. Remember last issue I gave you the hint that an easy way to do this is to put your machine language into memory (either by the IM command of EDTASM or by loading your assembled SYSTEM tape) and PEEK the location so the decimal values are on the screen while you write your DATA line, such as lines 4, 5, 13 in the above program.

    Then build a string composed of this DATA as line 9 above. You will see others don't build their string in this way. They assign the required space to the string, then determine where the string is, and POKE it in. While this works, and look at some of the back issues of MICRO-80 for an example, I (naturally) think my way is better in that it's easier to understand. But both methods do the job. CLEAR string space if necessary.

    Now comes the tricky part. Though I don't think it is difficult once you understand it. What's necessary is to provide the entry address of the machine language routine to USR(∅)(or whatever calling method you are using, and more of other ways later). The little understood function of VARPTR (meaning variable pointer) is used. And don't feel guilty for not understanding it - not many do from the explanation in the Level II Manual! What VARPTR does is to find out where the given variable is stored. And it will work with all types of variables, but in this case, we are intersted in where our just created string is living.

    So in line 1∅ I've used X as the variable for the value given by VARPTR. But with a string, the first value given is the length of the string (used by the LEN function) so actually it's the next 2 bytes that are needed. And these are not actually the start of the string, but a pointer to the start of the string. So we need to PEEK them, not use them directly. That is, use the value they are pointing at, not use them themselves. Yes, I know it may be somewhat confusing but think about it until it "clicks".

    These two locations are given LSB first MSB last, which is the way the Z-80 expects them, so in the way they are used in line 1∅ they are correct, but to actually calculate the address you would have to reverse them. More of this later.

    The addresses being pointed to are now POKEd into 16526-7, which is the location for USR(∅) to find out where to jump to in memory to enter your machine language program.

    Now a philosophical point. You will have heard in an earlier edition of MICRO-80 the article on the "garbage collector" which reorganises string usage when the space allocated by CLEAR (or the default 5∅) is getting full. So it could be that BASIC will relocate your string (and the machine language it contains) between accesses (but not during, since you won't

be in BASIC). So you will need to repeat line 1Ø before each access if you've used any string function. I haven't, so this is why in line 12 I go to line 11, not line 1Ø. But you will remember in the article on input checking, when I listed my routine for line input, I provided alternative lines to keep the routine in a string. But I did warn that theory says it may not work - actually it did work! And now you see why - the "garbage collector" may have moved the string because I used VARPTR once to find the location of the input string, and again to find the location of the machine language routine. And accessing any string function is the same as accessing a string - you run the risk of the "garbage collector" taking action. If you didn't understand this whole paragraph, don't worry, it's only an explanation for the less beginning beginner.

3. In a REM line. You will also see this technique used in earlier editions of MICRO-80, in my opinion where it is not really necessary.

My reason for using it here (lines Ø and 2) is that this particular routine needs an absolute location. That is one where it will remain, so the operating system knows where to find it. Because this particular routine modifies the operating system and is constantly being used - it's not like one which is accessed only at particular points in the program.

Line Ø provides enough space to hold the machine language code contained in lines 4 and 5. And it really does mean space, not TABs. After running this program you will see that line Ø fills up with what looks like garbage, but it is actually the machine language program which also happens to contain some control characters (only by chance do some of the machine language instructions happen to be the same as screen-control codes). The same thing happens with the string created by line 9. If you PRINT$, you will see what I mean.

Line 2 is the difficult one (if indeed anything here is difficult to understand). So stay with it. 16548-9 (4ØA4-5H) is the pointer to the start of BASIC. The reason for using the pointers, and not the actual start location, (42E9 for Level II) is to allow for disk, or any other machine language program (such as Level III) which is already loaded and has relocated the start of BASIC. So SM is the MSB, SL is the LSB of start of memory. Note the use of meaningful variables names still! And more later about MSB-LSB. Then the whole address is created in SA by multiplying SM by 256 and adding to it SL. Also 8 is added to get past the 5 bytes of pointers BASIC keeps at the start of every line. (Actually 5 would be enough to add but I added 8 for safety, just as I left more space in the REM line than the 98 bytes taken by the machine language routine).

The reason for calculating the whole address before adding 8, and not just adding it to the LSB is in case the LSB Happens to be something like 250, which would become 2 when 8 is added and the MSB would be incremented if done in the correct way. If the LSB only is considered then the address would be in error.

Then having obtained the address in SA where the machine language can be put, I recalculate the MSB and LSB in SM and SL, for later use in the program.

Now it's a simple matter of putting the machine language code into the space provided by the REM line. Note the for-next counter at the end of line 2 is SA+97, whereas there are in fact 98 items of data. This is the trap I warned of last episode: the first data item will actually be put into SA! You cannot have a Ø as one of the data items if you want to keep the machine language in a REM line. Line 3 is impossible to write without Eddy Paay's ROM Manual to tell you that 16405,16435-7 are in the keyboard driver routine. The first two must be fooled into thinking no keyboard is present until the routine is POKEd into memory, and its location is given to 16406-7.

Line 6 puts these first two locations right again, so that each time the operating system scans the keyboard it jumps to the machine language routine before acting on the results of that scanning of the keyboard. The last two locations are the pointer to the first line number for BASIC, which must be put right before RUNning the program if it has any GOTOs or GOSUBs. The reason they will be wrong is if you've POKEd them wrong as suggested in line 1 before saving this program. It's suggested so that a potential user cannot LIST the program before it's RUN, and then he can't LIST it after it's RUN because the routine will then be operating! So obviously, don't POKE them wrong in your own copy; and don't RUN before you save it (only because line Ø will fill up with apparent garbage).

After you've RUN it once, you can delete all but line Ø, but you can't save it as such, because the operating system still needs to be told where to find the machine language program (contained in line Ø) (by line 3) when it's nearly loaded.

HOW THE ROUTINES WORK.

I haven't space to explain how these two (break-disable, and sound) work here. Just wait for a future instalment when an explanation and the source code may be forthcoming. They've been included here simply to illustrate the connected points, not for their own right.

But until then, if you want to change the disabling of SHIFT@ to some other key disabling, change the 25th DATA item in line 4 from 96 to the ASCII value of whatever key you want to disable.

Also experiment with changing the value in USR in line 11 to something else and see how the tone generated changes. For example, in line 12 you could put before GOTO11, INPUT"NOTE(Ø-255)";N, and change the USR to read USR(N).

Remember how I said user experimentation is the key to true understanding.

TWO BYTE ADDRESSING.

The concept of addressing is simple once you understand - like everything! The whole point about it is that because we have up to 65,535 addressable memory locations in RAM, we need two bytes to address, since each byte can hold only 8 binary digits, that is, 256 decimal. So by combining two of these bytes we can get 256 multiplied by 256. So we must decide how we are going to identify which is the one to be multiplied by 256, and which is the one to be added to that result. By convention, we call the one which is multiplied the Most Significant Byte and the other the Least Significant Byte, usually referred to as MSB and LSB. So MSB multiplied by 256 and added to LSB gives the real address in decimal. Conversely, to get the MSB from the real address, take INT(address/256); to get LSB, subtract MSB multiplied by 256 from the whole address. Or a simpler way of getting LSB is to take the whole address AND 255, as line 2 above. Details of why this works are below.

The only other complication with the addressing is that the Z-80 takes the two bytes "the wrong way round" - that is, LSB first, MSB second. But it's only a case of remembering. If you look at an assembled source listing you'll see in the operand column the address is given "right way round", but in the assembled hex code in the second left column it's "wrong way round".

AND.

Here seems a good place to explain the use of AND above. It's not the AND/OR of BASIC such as IFA=B AND C=D, but the binary AND which compares bit patterns in the byte. Just the same is used in assembly writing, a knowledge of which is essential, for any but the simplest assembly program. The only thing here is that AND is used in BASIC, but it's still comparing bit patterns. The whole concept of AND in BASIC can be examined by trial print instructions in the command mode. For example, if you PRINT4AND8 you'll get Ø, but if you type PRINT4AND7 you'll get 4.

```
  4 = ØØØØØ1ØØ                        4 = ØØØØØ1ØØ
  8 = ØØØØ1ØØØ                        7 = ØØØØØ111
AND = ØØØØØØØØ                      AND = ØØØØØ1ØØ
```

This is because what is really happening is as illustrated at left. The computer is comparing bit patterns of the two numbers you specified (4 and 8) and setting answer bits only when it finds two ones set in that column. If you say OR instead of AND it will set answer bits if it finds one bit set in the column, regardless of what the other one is.

```
 4 = ØØØØØ1ØØ
 7 = ØØØØØ111
OR = ØØØØØ111
```

```
  4 = ØØØØØ1ØØ
  7 = ØØØØØ111
XOR = ØØØØØØ11
```

XOR means exclusive OR, and will set a bit if it finds one and only one bit set in the column. This is useful if you want to set the accumulator to Ø, but not by loading it with Ø, because you are keeping it in a REM line which would interpret Ø byte to be the end-of-line marker.

THE STACK.

Being familiar with machine language, you will know of the stack which is used as a temporary storage area in machine language programming. So it is in BASIC too. It normally locates itself 55 bytes below the top of memory (or protected memory). This is the 50 bytes default string space, plus the 2 bytes extra set aside from very top of memory, plus 2 bytes separating string space and stack space. The stack grows downwards (high memory address to low). However, as said last episode, you don't need to worry too much about conflicting with the stack unless you are putting your machine language into memory and setting memory size from within a GOSUB or middle of a complex BASIC line. Of course, your machine language should have balanced its PUSHes and POPs before it returns to BASIC, but so it should anyway.

PROTECTING MEMORY.

By answering the memory size question, you're really specifying the highest memory location BASIC can use. Actually it will use two locations under what you've set. If you look at a memory map you'll see that immediately under the top of memory is the string space, which BASIC uses to work on any strings from your program. So this is what will overwrite your machine language routine if you don't set memory protection. Of course, if you've kept it in a string or REM line you won't need to protect memory.

POINTERS.

You will notice in the above program that I've used pointers rather than absolute addresses. This is a very convenient way of writing as it allows your program to be portable from your own machine to other machines of different memory and equipment configurations.

I won't go into the huge task of detailing the pointers here. Eddy Paay's ROM Manual does though, so I suggest you look there.

Also in the ROM Manual you'll find the entry for other ways to get into machine language without using USR(Ø). And others are useful in that disk BASIC has a different syntax for USR, so your program would need to be changed for a disk machine if it uses USR. However, the others are not as easy to pass variables to and from the machine language routine and BASIC although it can be done in several other ways. An example is to POKE the LSB and MSB into 16789-9Ø instead of 16526-7. This will cause a jump to the machine language when the computer encounters the & which is a disk BASIC command, and has its entry point at 16789-9Ø.

If, like me, you think this is really rivetting stuff, you will certainly be well rewarded by the ROM manual.

WHY TALK HEX.

This is an often asked question. I used to ask it myself but when you get really "into" machine language and develop a "feel" for it, you will realise why and I'm sorry that's all I can give you for a reason. But so many (me included) do. I hope you will just take my word for it. I could tell you it's easier to remember a 4 digit number than a 5 digit decimal one, but you probably won't accept that. Just remember my words when you are an accomplished hex talker and somebody asks you why.

MONITORS-UTILITIES.

You'll read of various machine language programs, some of which are referred to as monitors and some as utilities. It seems there is some overlap, but I consider a monitor to be one which allows you to modify a machine language program, set break points, single step, etc. such as ZBUG. I consider a utility to be one which allows you to do things other than fix a machine language program. So to my mind, most programs about are utilities. And the one I use is MON3 (MON4 is available for disk). Eddy Paay has combined MON3 with a high memory version of TBUG to give the best of both worlds. MON3 is produced by Howe Software, and is available in Australia from Computer Campus at Kent Town, S.A.

Another I use is PACKER, but this is obviously not in the same area. Although ZBUG (which comes with Microsoft Editor/Assembler Plus) is a really good and powerful monitor, I can't honestly recommend it because it's so difficult to use - even Eddy says so! But as EDTASM, which it comes (free?) with is so good, it doesn't really matter.

Therefore if you want to know what I'd suggest you get to help in combining your machine language with your BASIC (though often you won't need anything) and to allow you to "fiddle" with others' machine language, I'd recommend MON3. Admittedly it has its limitations, as do most, but then what are the chances that you'll find any to suit your requirements exactly! I do get by quite well with MON3 most of the time because fiddling about with others' machine language is what I like best.

CONCLUSION.

Still to come are sound routines with a source listing and explanation, and details of break key handling with a source listing. I haven't space to detail or explain the two routines listed above so this will come next time. And also to come is a function simulating ON BREAK GOTO. I hope the above has been of value. Remember my constant preaching that nothing is a better teacher than experience, practice and thought.

At the end of this series there will be an episode devoted to answering readers' questions. So please send some in.

AN EXPLANATION - by Rod Stevenson

A number of readers have assumed from my references to the Adelaide Users' Group that it is the same as MICRO-80. I want to point out that the two are in no way connected. The only common factor is that both are in Adelaide; therefore Editor, Ian Vagg, attends meetings and gets ideas from speakers at the Group. This is how I came to be writing this series, and how Allan Dent's articles on the joystick and port project appear. Other members of the Group (Bernie Simson, Bruce Bussenschutt, etc.) just happen to write and submit articles just as many others from all over Australia do. (What he means is, we are just good friends!! - Ed.)

- 0000000000 -

***** JOYSTICKS AND INPUT/OUTPUT PORTS FOR YOUR '80   by Alan Dent *****

PART 3 - A/D - I/O BOARD ASSEMBLY INSTRUCTIONS AND DETAILED PPI PROGRAMMING DESCRIPTION

Firstly I cannot overstress the importance of a low wattage soldering iron with a small tip. If your iron is not one of the small pencil type and has an unplated copper tip, file the end to a small surface preferably about 1/16th and no larger than 1/8th of an inch tip width. Keep the tip clean, tinned and wipe it regularly with a piece of cloth during construction. Alternatively, you may be able to borrow a more suitable iron from a friend. Failure to use a suitable iron can be disastrous by causing tracks to lift and creating solder bridges etc. You may even be able to use this as a good reason to upgrade your old iron. A small pair of clean cutting wire cutters will also be required, and a pair of long nose pliers would be helpful.

The easiest way to assemble the P/C board is to start with the wire links joining the tracks through the board. As mentioned in the last instalment, for reasons of economy, the board does not have plated through holes. First, place some spacers about the thickness of a match on your bench and lay the P/C board on top of them, component side uppermost. An alternative method would be to temporarily fasten some matches to the underside of the P/C board with masking tape, taking care not to cover any holes to be used for the wire links. Insert the wire through the hole linking the tracks so that the wire touches the benchtop. Solder the wire in place and then cut the wire just above the solder joint, do not solder the other side at this point in time. Repeat this process for all tracks which have holes in them with the following exceptions:

1) integrated circuits
2) top soldered resistor between IC2 and IC3
3) capacitor between IC3 and IC4 (middle hole in track)
4) capacitor between the PPI's, IC8 and IC9
5) the optional capacitors near the port connectors
6) -ve connection at the board edge near the regulator
7) corner hole for the PORT 13 connector

When finished soldering-in all the links on the top of the board, turn it over and lay it flat on the bench. Bend all the leads back along their respective tracks, flat against the P/C board and solder them. Some links may be too long to do this and will have to be cut to a more suitable length.

Next, solder in the IC sockets with pin 1 as indicated. Do them one at a time as it is much easier to hold only one socket in place while turning the board over to solder it in. With the sockets in place, you now have reference points which make it easier to locate the holes for the other components. TRS-80 owners will not have to use IC10; this is only used by System-80 users. The resistors should be installed next and then the small capacitors. Don't mount the components flush with the P/C board but raised up so that the resistors are about the same height as the IC sockets. This allows for easier servicing if it should ever become necessary (heaven forbid). The extra lead length allows a CRO probe to be easily clipped on for monitoring the various pulses. This is a good habit to get into whenever you construct any electronic device, space permitting of course (this is written by an electronic serviceman in case you haven't guessed). Be careful to insert the 10μf correctly polarised. Now instal the bridge rectifier, polarised as indicated on the underside of the board, then the 2000μf similarly polarised. Finally the regulator IC polarised as shown in the stuffing diagram followed by the PORT and A/B connectors and the 4 pins for the AC/DC connections next to the rectifier. System 80 owners will also have to fit a 3 pin male connector alongside IC10. Orient the board to place the port connections at the top, the connections for this 3 pin connector from top to bottom are: READ - WRITE - I/O REQUEST.

As the board was originally designed for a TRS-80 model 1, model 1 owners may now insert the 40 pin header plug into the BUSS CONNECTOR socket, the ribbon cable feeding out over the PPI port side of the board. Non model 1 owners who are anxious to construct this peripheral will have to do it the hard way as described later in the text, but for those not in such a hurry, MICRO-80 are working on an adaptor which will allow you to directly connect it to a ribbon cable from the computer expansion port. Model III owners will have to strip back the ribbon cable and solder the various wires to the header pins from the connection information supplied. The wires to be connected are the 8 data lines, the lower 8 address lines and the control lines WAIT, IN, OUT and RESET. System 80 owners will also have to do as the Model III owners did plus you will have to generate the IN and OUT signals in IC10. To do this the I/O request line and the READ and WRITE lines have to be combined in IC10 by connecting them to the 3 pin connector as indicated on the stuffing diagram and also mentioned in a previous paragraph.

Now for the power connections. The circuit is designed for either A/C or D/C input. Those who are going to use A/C from a low voltage transformer of about 8 to 12 volts will use the connections indicated by the sinewave on the underneath of the board. You also have the advantage of being able to tap off the unregulated D/C if required. Those using an external D/C supply such as a "plug pack" type should use the D/C pins correctly polarised as indicated with the + and - signs. A 9 volt battery snap connector such as TANDY 270-235 or DICK SMITH P-6216 is quite suitable for this. Connect your power pack to the connector, check that it is switched to 9 volts and the polarity connector is + to + and - to -. If you have a multimeter you can check the polarity for yourself before you connect it to your precious board. You may also like to monitor the 5 volt supply as you switch on the power, or even check the regulator before

you plug in the other chips.

Now insert the I/c's in their sockets. TTL first (practice with the cheap ones). If the pins
are spread too wide for the socket, gently bend each row of pins inwards using a pair of long-
nose pliers being careful not to overdo the job. When handling the LSI chips the normal static
electricity precautions should be observed. Don't remove them from their static protected package
until you have finished everything else and you are ready to fit them. Earth yourself and the
board to ensure that all static electricity is discharged; remove the LSI chips and insert them
in their respective sockets. Although most MOS LSI chips have static protection built in these
days, it is still good practice to observe the above precautions. For those of you who are
worried about earthing everything while living in the modern vinyl/synthetic house, the kitchen
sink is an excellent place to do this. Those of you who have traded in your sink on a dish
washer have a slight problem!

Now comes the real test!!! Switch on the power to the board without it being connected to your
computer. If, when you switch on, everything just sits there calmly, very good. You may now
proceed to the next step, but if the power pack hums loudly and/or smoke pours from the board,
first panic and then switch off and try to find out what went wrong. Hopefully this will not
happen to anybody. Next switch off everything, connect your ribbon cable to your computer edge
connector. Plug in the cable that connects your JOYSTICKS to the A/B inputs, switch on the
plug pack first, followed by the computer. Type in your simple test program and run it, call
your family and friends and show them what a clever boy you are.

Now for some more detailed programming information on the operation of the board. The A/D input
is straight forward in that it is addressed as an input port exactly as described in the LEVEL
II handbook. It is also addressed in assembly language as an input port. In both cases, the
8 bit value of the analogue input voltage of the addressed port is returned to the computer.
In the case of the LEVEL II program, the 8 bit value is returned in decimal value to the variable
used in the input statement. The assembly program input command returns the digital value directly
to the accumulator. The BASIC command would be in the form of Variable=INP(Port number)
e.g. X=INP(Ø). Assembly language programmers would use IN A(Port number).

As you can see, using the A/D function is fairly straight forward but programming the Parallel
I/O Ports is a little more complicated. The Programmable Peripheral Interface (PPI), INTEL
or NATIONAL SEMICONDUCTOR 8255 is capable of 3 modes of operation. The first of these is Mode
Ø, which is also the simplest to use. It allows data to be written to or read from any of the
available ports (A, B or C) without the use of any "handshaking" signals. Before any data is
transferred to or from the PPI, it must first be "set up" or programmed. As in any computer
operation, the use of the PPI must be pre-planned into the program using the device. With this
knowledge of the PPI requirements, we know in advance in which "direction" we want the ports
programmed. As an example, we may require 8 input or sense lines and 16 output or control lines.
This requirement can be met in 3 different ways, A IN with B and C OUT, or B IN with A and C
OUT, or C IN with A and B OUT. As you can see the PPI is very adaptable, but how do we "set
up" this chip to do what we want? The PPI is programmed by writing a CONTROL WORD into the
CONTROL REGISTER. Sounds simple doesn't it - well it is. If you look at the Mode Ø Port Definition
Chart, a quick scan down the Group A - Group B columns will reveal the 16 possible combinations
of input and output ports. Remember from last month, port C can be split into 2 groups of 4
lines giving in effect 4 ports. Looking at these columns, we can pick out the control word
to suit our requirement. The control words to fit the 3 variations mentioned above are 144,
13Ø, 137, respectively or, if you prefer, HEX 9ØH, 82H, 89H. If you're a binary freak, you
will notice a tie-up between the control word BIT columns and the 4 port columns. Data bit
4 controls the direction of port A, a "1" for IN and a "Ø" for OUT. Data bit 3 similarly controls
the upper 4 bits of port C, data bit 1 controls port B and data bit Ø controls the lower 4 bits
of port C. Now that you know what control word is needed, it has to be written into the control
register. Looking at the Port Select chart, we see that the control register sits at chip address
3. Our PPI's are placed at addresses 8 and 12 (IC9 and IC8 respectively). This means that
their control registers are located at 3 address locations above this, or Port addresses 11
and 15. If we want to set both PPI's to the same condition, e.g. A IN, B and C OUT, we would
write to both control registers at addresses 11 and 15, the control word 144 decimal or 9Ø HEX.
On receiving the new control word, the PPI clears all internal registers and sets the various
ports to the direction specified by the control word. If we now request an input from port
A, (IN port 8 or IN port 12), the 8 bit word being presented to the ports external pins is latched
into the A register and then placed on the data bus for the computer. To write to the ports
is equally simple - to output to port B, simply OUT 9,XXX or OUT 13,XXX and the value is latched
into the output latch of port B. Port C is similar except that it uses port addresses 1Ø and
14 depending on which chip you use. At this point I should correct a typing error in the second
part of this article. In the fourth line from the bottom of page 10, the reference to "PORT
B" should be changed to "PORT 8". I hope this clears up any confusion that may have been caused.

The output of the PPI port C can also be varied by the BIT SET/RESET control word. If you look
at the diagram you will see how to select the control word to perform your required function.
The bit set/reset function is selected by data bit 7, a "Ø" selects the bit set/reset mode.
The next 3 bits D6,D5,D4 are not used and should be Ø, bits D3,D2,D1 select in binary, the bit
that you want to change while bit DØ selects whether you want to set it "1" or reset it "Ø".
As an example we will set data bit 7 on port 10. To do this the control word is as follows:
D7=Ø (selects bit mode) D6,D5,D4-Ø (not used) D3,D2,D1=1 (selects data bit 7)   DØ=1 (sets the

bit), this gives us the control word 15 or 0F Hex and Bit 7 is set regardless of its previous condition while the remaining bits remain unchanged.

The remaining modes 1 and 2 will not be covered here as they do not generally apply in the type of design presented, although I'm sure someone could if the need arose. If you want to use these modes, I suggest that you consult the manufacturer's data books for details. In the future if a project that uses these other modes is developed by somebody and submitted to MICRO-80, I am sure that a detailed description of these modes of operation will be included with the article.

8255 PROGRAMMABLE PERIPHERAL INTERFACE - DATA

mode 0 port definition chart

DECIMAL No	HEX No.	Control Word Bits								Group A		Group B	
		D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Port A	Port C (Upper)	Port B	Port C (Lower)
128	80	1	0	0	0	0	0	0	0	OUTPUT	OUTPUT	OUTPUT	OUTPUT
129	81	1	0	0	0	0	0	0	1	OUTPUT	OUTPUT	OUTPUT	INPUT
130	82	1	0	0	0	0	0	1	0	OUTPUT	OUTPUT	INPUT	OUTPUT
131	83	1	0	0	0	0	0	1	1	OUTPUT	OUTPUT	INPUT	INPUT
136	88	1	0	0	0	1	0	0	0	OUTPUT	INPUT	OUTPUT	OUTPUT
137	89	1	0	0	0	1	0	0	1	OUTPUT	INPUT	OUTPUT	INPUT
138	8A	1	0	0	0	1	0	1	0	OUTPUT	INPUT	INPUT	OUTPUT
139	8B	1	0	0	0	1	0	1	1	OUTPUT	INPUT	INPUT	INPUT
144	90	1	0	0	1	0	0	0	0	INPUT	OUTPUT	OUTPUT	OUTPUT
145	91	1	0	0	1	0	0	0	1	INPUT	OUTPUT	OUTPUT	INPUT
146	92	1	0	0	1	0	0	1	0	INPUT	OUTPUT	INPUT	OUTPUT
147	93	1	0	0	1	0	0	1	1	INPUT	OUTPUT	INPUT	INPUT
152	98	1	0	0	1	1	0	0	0	INPUT	INPUT	OUTPUT	OUTPUT
153	99	1	0	0	1	1	0	0	1	INPUT	INPUT	OUTPUT	INPUT
154	9A	1	0	0	1	1	0	1	0	INPUT	INPUT	INPUT	OUTPUT
155	9B	1	0	0	1	1	0	1	1	INPUT	INPUT	INPUT	INPUT

Port Select (A_0, A_1): These two inputs, which are normally connected to the least significant bits of the A_{15} - A_0 Address Bus, control the selection of one of three 8-bit ports (A, B and C) or the internal control word register as indicated below.

A_1	A_0	Selected
0	0	Port A
0	1	Port B
1	0	Port C
1	1	Control Word Register

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

MODE SET FLAG: 1 = ACTIVE

MODE SELECTION: 00 = MODE 0 / 01 = MODE 1 / 1X = MODE 2

PORT A: 1 = INPUT 0 = OUTPUT

PORT C (UPPER): 1 = INPUT 0 = OUTPUT

MODE SELECTION: 0 = MODE 0 1 = MODE 1

PORT B: 1 = INPUT 0 = OUTPUT

PORT C (LOWER): 1 = INPUT 0 = OUTPUT

GROUP A   GROUP B
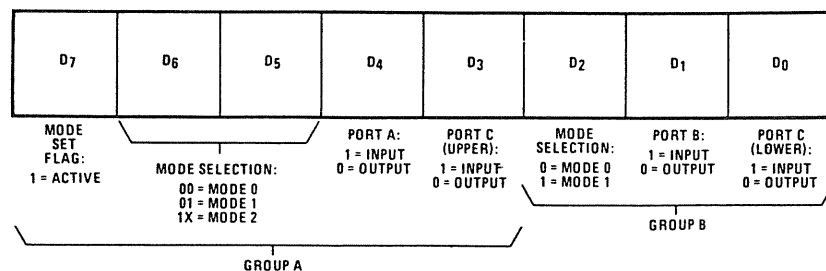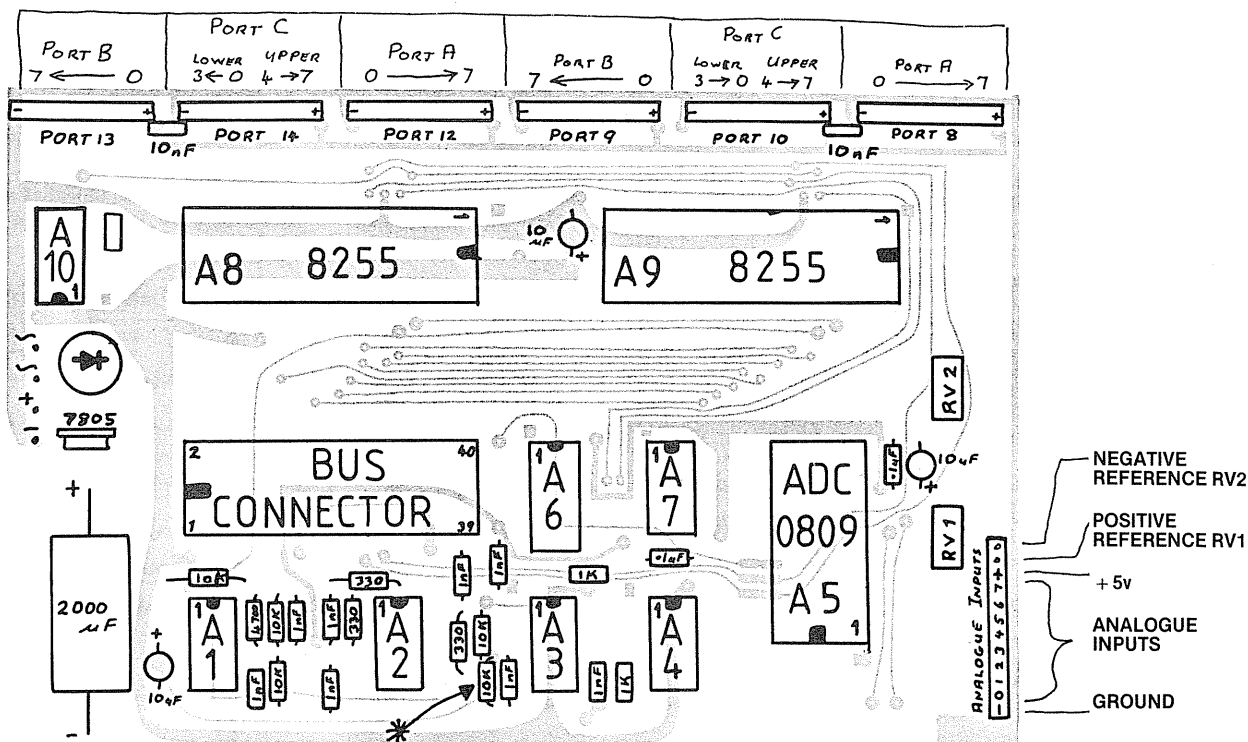
mode definition control word format

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

BIT SET/RESET FLAG: 0 = ACTIVE

NOT USED

BIT SELECT:
000 = 0
001 = 1
010 = 2
011 = 3
100 = 4
101 = 5
110 = 6
111 = 7

BIT SET/RESET: 1 = SET 0 = RESET
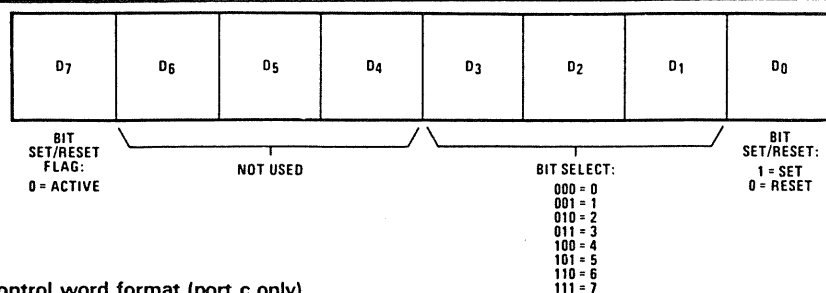
bit set/reset control word format (port c only)

COMPONENT STUFFING DIAGRAM

PLEASE TAKE NOTE OF THE PORT PIN NUMBERING
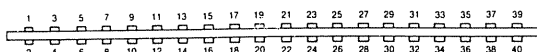
This is the only top soldered component. Unless the
optional 10nF capacitors alongside the parallel ports
are fitted, they have to be top soldered also.

TRS-80 EXPANSION PIN EDGE

P/N	SIGNAL NAME	DESCRIPTION
1	RAS*	Row Address Strobe Output for 16-Pin Dynamic Rams
2	SYSRES*	System Reset Output, Low During Power Up Initialize or Reset Depressed
3	CAS*	Column Address Strobe Output for 16-Pin Dynamic Rams
4	A10	Address Output
5	A12	Address Output
6	A13	Address Output
7	A15	Address Output
8	GND	Signal Ground
9	A11	Address Output
10	A14	Address Output
11	A8	Address Output
12	OUT*	Peripheral Write Strobe Output
13	WR*	Memory Write Strobe Output
14	INTAK*	Interrupt Acknowledge Output
15	RD*	Memory Read Strobe Output
16	MUX	Multiplexor Control Output for 16-Pin Dynamic Rams
17	A9	Address Output
18	D4	Bidirectional Data Bus
19	IN*	Peripheral Read Strobe Output
20	D7	Bidirectional Data Bus
21	INT*	Interrupt input (Maskable)
22	D1	Bidirectional Data Bus
23	TEST*	A Logic "0" on TEST* Input Tri-States A0-A15, D0-D7, WR*, RD*, IN*, OUT*, RAS*, CAS*, MUX*
24	D6	Bidirectional Data Bus
25	A0	Address Output
26	D3	Bidirectional Data Bus
27	A1	Address Output
28	D5	Bidirectional Data Bus
29	GND	Signal Ground
30	D0	Bidirectional Data Bus
31	A4	Address Bus
32	D2	Bidirectional Data Bus
33	WAIT*	Processor Wait Input, to Allow for Slow Memory
34	A3	Address Output
35	A5	Address Output
36	A7	Address Output
37	GND	Signal Ground
38	A6	Address Output
39	+5V	5 Volt Output (Limited Current)
40	A2	Address Output

NOTE: *means Negative (Logical "0") True Input or Output

Mates with AMP P/N 88 103-1 Card
Edge Connector or Equivalent
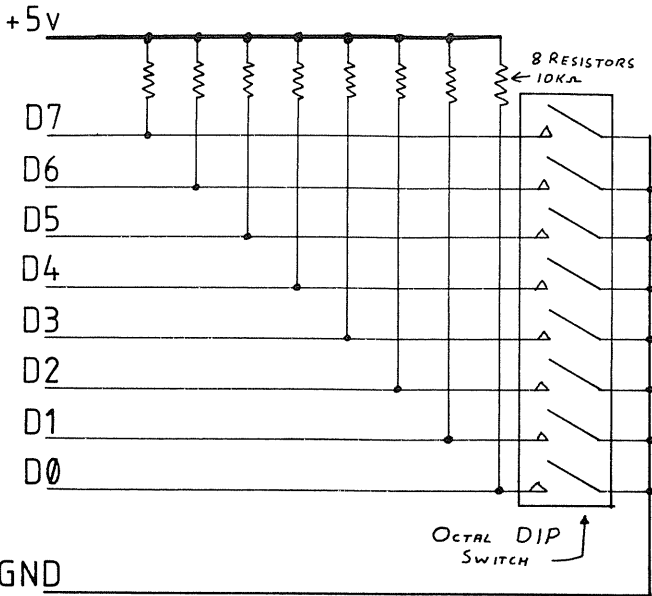
Connection points for Expansion-Port Edge Card
(viewed from rear of keyboard assembly)

SYSTEM-80 EXPANSION PIN EDGE
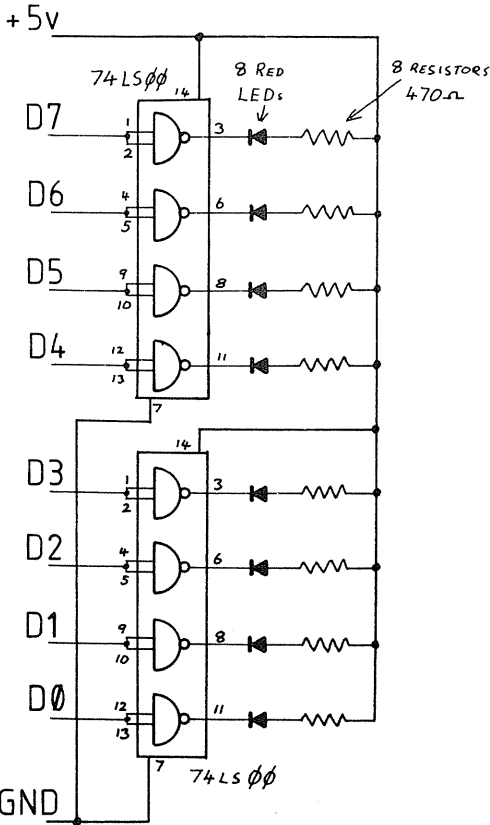
VIEWED FROM REAR SIDE

PIN	SIGNAL		
1	GND	26	A10
2	GND	27	A13
3	A7	28	A11
4	A6	29	A12
5	A5	30	PHI
6	A4	31	PINT
7	A1	32	NC
8	A3	33	NC
9	A2	34	PHLDA
10	A0	35	PHANTOM
11	D5	36	HALT
12	D2	37	PWAIT
13	NC	38	IORQ
14	D1	39	PHOLD
15	D0	40	WR
16	D3	41	RD
17	D7	42	CCDBS/STADBS
18	D6	43	MREQ
19	VCC	44	DODBS/ADDBS
20	D4	45	M1
21	A15	46	RESET
22	A8	47	RFSH
23	A14	48	NMI
24	A9	49	GND
25	NC	50	GND

SIGNAL EXPLANATION

PHI	1.79MHz CLOCK
PINT	INTERRUPT
NC	NO CONNECTION
PHLDA	PROCESSOR HOLD ACKNOWLEDGE
PHANTOM	PHANTOM
HALT	HALT ACKNOWLEDGE
PWAIT	PROCESSOR WAIT
IORQ	INPUT/OUTPUT REQUEST
PHOLD	PROCESSOR HOLD
WR	PROCESSOR WRITE
RD	PROCESSOR READ
CCDBS/STADBS	CONTROL AND DATA BUS DISABLE
MREQ	MEMORY REQUEST
DODBS/ADDBS	DATA AND ADDRESS BUS DISABLE
M1	FIRST STATE OF INSTRUCTION CYCLE
RESET	CPU RESET
RFSH	DYNAMIC MEMORY REFRESH
NMI	NON-MASKABLE INTERRUPT

+5V

8 RESISTORS
← 10KΩ

D7
D6
D5
D4
D3
D2
D1
D0

GND

Octal DIP Switch

PORT INPUT SWITCH

+5V

74LS00   8 RED LEDs   8 RESISTORS 470Ω

D7
D6
D5
D4
D3
D2
D1
D0

GND

74LS00

PORT OUTPUT MONITOR

SYSTEM-80 TO TRS-80

RD

IORQ   IN

WR   OUT

74LS32

240V/9-12V @ 1A

LM309

+5V

2000µF 16V

Gnd
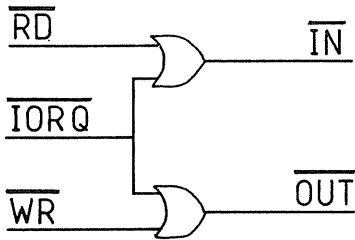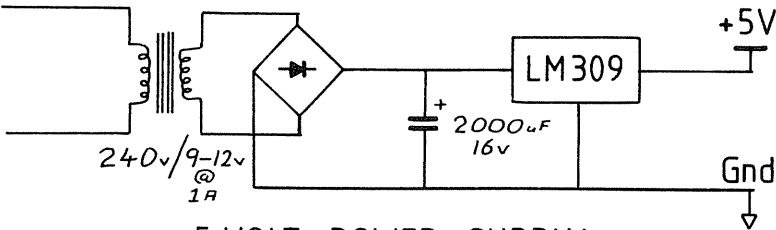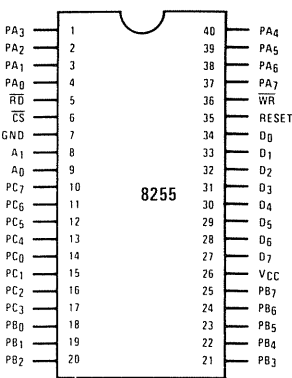
5 VOLT POWER SUPPLY

8255A BASIC OPERATION

A$_1$	A$_0$	RD	WR	CS	INPUT OPERATION (READ)
0	0	0	1	0	PORT A ⇒ DATA BUS
0	1	0	1	0	PORT B ⇒ DATA BUS
1	0	0	1	0	PORT C ⇒ DATA BUS
					OUTPUT OPERATION (WRITE)
0	0	1	0	0	DATA BUS ⇒ PORT A
0	1	1	0	0	DATA BUS ⇒ PORT B
1	0	1	0	0	DATA BUS ⇒ PORT C
1	1	1	0	0	DATA BUS ⇒ CONTROL
					DISABLE FUNCTION
X	X	X	X	1	DATA BUS ⇒ 3-STATE
1	1	0	1	0	ILLEGAL CONDITION
X	X	1	1	0	DATA BUS ⇒ 3-STATE

pin configuration

		8255		
PA$_3$	1		40	PA$_4$
PA$_2$	2		39	PA$_5$
PA$_1$	3		38	PA$_6$
PA$_0$	4		37	PA$_7$
RD	5		36	WR
CS	6		35	RESET
GND	7		34	D$_0$
A$_1$	8		33	D$_1$
A$_0$	9		32	D$_2$
PC$_7$	10		31	D$_3$
PC$_6$	11		30	D$_4$
PC$_5$	12		29	D$_5$
PC$_4$	13		28	D$_6$
PC$_0$	14		27	D$_7$
PC$_1$	15		26	V$_{CC}$
PC$_2$	16		25	PB$_7$
PC$_3$	17		24	PB$_6$
PB$_0$	18		23	PB$_5$
PB$_1$	19		22	PB$_4$
PB$_2$	20		21	PB$_3$

Joysticks and I/O Ports (cont.)
We will conclude this series of articles with some sample programs in both BASIC and machine language to illustrate some of the possible applications of this hardware project. The final part will also include full-sized PC board masters for the TRS-80 Model I. At a later date we will describe how to construct separate adaptor boards for SYSTEM-80 (Video Genie) and Model III computers. It is our intention to make the PC boards available to our readers to assist with the project. The price of the board will be announced in a future issue.

-0000000000-
***** '80 USERS' GROUPS *****

The following is a list of '80 Users' Groups. If you have a group that is not included here, please let us know about it so that we can publish details. Owners of System '80s are welcome at all the groups.

** AUSTRALIA **

AUSTRALIAN CAPITAL TERRITORY

CANBERRA GROUP

Contact: Bill Cushing, 10 Urambi Village, Kambah, ACT. 2902.
MEETINGS: 3rd Thursday of each month at 7.30 p.m. in:
          Urambi Village Community Centre, Crozier Circuit Kambah.

NEW SOUTH WALES

COMPUTERTOWN CAMDEN (CTAUS!)

Contact: Keith Stewart
          P.O. Box 47, Camden, N.S.W. 2570.

SYDNEY MAPPER CP/M GROUP

Contact: Dan Lawrence, Mapper CP/M Users' Group,
          c/o G.P.O. Box 2551, SYDNEY, N.S.W. 2001.

SYDNEY EASTERN SUBURBS

Contact: Dan Lawrence,
          TRS-80 Eastern Suburbs Group,
          c/o G.P. Box 2551, SYDNEY, N.S.W. 2001.

WOLLONGONG GROUP

Contact: Paul Janson,
          P.O. Box 397, DAPTO, N.S.W. 2530.

NORTHERN TERRITORY

DARWIN GROUP

Contact: Tony Domigan,
          P.O. Box 39086, WINNELLIE, N.T. 5789.

QUEENSLAND

BRISBANE GROUP

Contact: Lance Lawes,
          Tel: Home (07)396 2998
               Bus. (07)268 1191 Ext. 15
MEETINGS: 1st Sunday each month at 2 p.m. at 21 Rodney Street, Lindum.

SOUTH AUSTRALIA

ADELAIDE GROUP

Contact: Rod Stevenson,
          36 Sturt Street, ADELAIDE, S.A. 5000.
          Tel: 51 5241 bet. 9-4.

VICTORIA

EASTERN SUBURBS GROUP

Contact: Mr. John Fletcher
          Tel: Home (03)737 9544
               Bus. (03)89 0677 (9-4)
MEETINGS: 4th Wednesday of the month at 7 p.m. at
          Kingswood College, 355 Station Street, Box Hill

MICOM 80 USERS' GROUP

Contact: Len Sanders
          Tel: (03560 8132
MEETINGS: 1st Wednesday of each month at Alvie Hall,
          High Street, Mt. Waverley.

NORTHERN AND WESTERN SUBURBS

Contact: Mr. David Coupe
          Tel: (03)370 9590
MEETINGS: C.P.M. Data Systems, 284 Union Road, Moonee
          Ponds - Alternate Thursdays at 7 p.m.

PENINSULAR GROUP

Contact: M.G. Thompson - (03)772 2674
MEETINGS: 2nd Tuesday of the month (except Jan.)

GEELONG COMPUTER CLUB

Contact: The Geelon Computer Club,
          P.O. Box 6, GEELONG, VIC. 3220.
MEETINGS: 2nd Tuesday of the month at Tybar Engineering
          Hampton St. Newton.

** UNITED KINGDOM **

NATIONAL   NATIONAL USERS' GROUP

Contact: Brian Pain, 40 High Street, Stoney Stratford,
          Milton Keynes.

CO. DURHAM   NORTH-EAST TRS-80 GROUP

Contact: Barry Dunn, 8 Ethick Tce. North Craighead,
          Stanley, Co. Durham, DH9 6BE. Tel: 0207-30184

COMPUTERTOWN UNITED KINGDOM (CTUK!)

Contact: Dave Tebbutt, c/o 14 Rathbone Place, London
          W1P 1DE

TANDY OWNERS PROGRAM & INFORMATION CO-OP.

Contact: Derek Higbee, 12 Shelley Close,
          Ashley Heath, Ringwood.
          Tel: Ringwood 6720.

NATIONAL   NEWCASTLE PERSONAL COMPUTING SOCIETY

Contact: John Stephen Bone - 0632 770036

BOLTON   NORTH-WEST TRS-80 GROUP

Contact: The Secretary, North-West TRS-80 Users Group,
          40 Cowlees, West Houghton, Bolton BL5 3EG.

COMPUTERTOWN NORTH-EAST (CTNE!)

Contact: c/o 2 Claremont Place, Gateshead, Co. Tyne &
          Wear NE8 1TL. Tel: 0632-770036/643417/679119/
          559167

***** MICRO-BUGS *****

In which we correct those errors which seem to creep in, no matter how careful we are.

SUPER SIZZLER - Issue 6, pages 39-43.

There are two bugs in this program:

(a) upon trading in the chassis, the amount of the debt is indeed decreased by $750 dollars, but in fact the jacks are removed.

(b) Upon trading in either the wheels or the body, the correct part is removed, but the debt is increased rather than decreased, with the result that the program enters an endless loop, with the poor player's indebtedness increasing without limit.

The following amendments to the program will remove these difficulties:

(a) Line 1010 - make this read "IF PS=-1 THEN 1130"

(b) New line 1011 - "X=41: FOR Y=10 TO 13 etc." (to the end of the old line 1010)

(c) Line 1130 - change this to: "FOR X = 208 TO 237:PRINT @ X, " ";: NEXT: GOSUB 1011: RETURN" (note the change to the GOSUB address).

(d) Line 1370 - add the instruction "PS=-1" immediately after "GOSUB 1200".

(e) Line 1450 - once again, add "PS=-1" immediately after "XX = 0".

Some additional changes necessary to make the program perform sensibly on a "SYSTEM 80" are as follows:

(a) Line 320 - change CHR$(94) to CHR$(62) and CHR$(93) to CHR$(60) - (in two places for each).

(b) Lines 520, 590, 600, 630 and 640 - change "[" to CHR$(168).

(c) Lines 570 and 580 - change CHR$(8) to CHR$(44) and CHR$(9) to CHR$(46).

(d) Line 2130 - add ELSE CLS immediately following RUN.

(e) Line 130 - change to read: ".......MONEY. USE THE LEFT AND RIGHT ARROW KEYS ";

ASTRONOMY 1.0 - Issue 15, pages 21-31.

There has been considerable interest in this program and many readers have contacted the author directly. He reports that Lines 811, 2320, 4036 and 4110 should be changed to remove a problem in section 3 with rising and setting of fixed objects and to improve accuracy. Edit the lines to read as follows:

```
811  INPUT "ENTER DAY, MONTH, YEAR";D,MO,Y:F1=D:F2=MO
2320 GOSUB 7200
4036 V1=V : V=V+282·510396
4110 T=COS(V1*J1)*0·016720+1 : T2=0·9997156 : F=T/T2
```

U.K. reader, Reg Fidler, points out a typographical error in the text accompanying the program. In paragraph 3 (Greenwich Sidereal Time to Greenwich Mean Time conversion) the answer is given as 12 hours 57 mins. 59 seconds. This should be 12 hours 51 mins. 59 seconds.

Reader, R.J. Sykes, suggests an error trapping routine to overcome an "FC ERROR" which can appear on one calculation.

```
15105 ON ERROR GOTO 19000
19000 QP=ERR/2+1:IF QP=5 THEN PRINT "THIS OBJECT EITHER DOES NOT RISE OR SET AT YOUR LOCATION."
19010 PRINT "CHECK YOUR CO-ORDINATES OR CHOOSE ANOTHER OBJECT."
19020 RESUME 18000
```

UTIL 1, 2, 3 & 4 - Issue 21, pages 28-34.

During the printing of these program listings some lines did not reproduce.

UTIL 1

```
4315    28 11        JR  Z,4328     READY
4325    C3 58 04     JP  0458       Return to ROM
4331    CD 65 43     CALL 4365      FLASH
433A    32 83 43     LD(4383),A     Save the character to repeat
4345    F2 42 43     JP P, 4342     KLOOP
434B    2A 81 43     LD HL,(4381)   Save repeat count
4357    32 7F 43     LD(437F),A     Reset counter
435D    3A 83 43     LD A, (4383)   Fetch the character to repeat
```

```
4360    22 81 43        LD(4381),HL     Adjust the repeat counter
4366    2A 7F 43        LD HL,(437F)    Fetch the flash count
4375    21 00 03        LD HL, 0300     Flash rate delay
4378    22 7F 43        LD (437F),HL    and save it
438B    21 30 43        LD HL, 4330     point to REPEAT
4391    21 85 43        LD HL, 4385     point to end +1
                                        change 85 to AA to save INIT.
4398    22 A4 40        LD (40A4),HL
```

UTIL 2

Some instructions had the high byte 43 missing. The address of these instructions are:

```
4F55    4F5B    4F65    4FAC    4FB5    4FC0    4FC6
4FD2    4FD8    4FDB    4FE1    4FF3
```

The 43 byte should be in the second address following the above addresses.

UTIL 3

```
42EA    21 A9 43        LD HL, 43A9     scroll on/off flag
4315    28 10           JR Z, 4327      READY
4325    18 08           JR, 432F        LWRCSE
4356    CD 8A 43        CALL 438A       FLASH
435F    32 A8 43        LD (43A8),A
436A    F2 67 43        JP P, 4367      KLOOP
4370    2A A6 43        LD HL, (43A6)
4382    3A A8 43        LD A, (43A8)
4385    22 A6 43        LD (43A6), HL
438B    2A A4 43        LD HL, (43A4)
439A    21 90 01        LD HL, 0190
439D    22 A4 43        LD (43A4), HL
43B0    21 55 43        LD HL, 4355
43B6    21 AA 43        LD HL, 43AA     change AA to CF to save INIT.
```

UTIL 4

High byte 4F missing from the following instructions -

```
4F30    4F36    4F40    4FAC    4F85    4FC0
4FC6    4FD2    4FD8    4FDB    4FE1    4FF3
```

The flashing cursor may leave unwanted characters on the screen during graphic displays. This may be eliminated by turning the cursor off.

```
Util    Address
1       17267
2       20462   32750   4K & 16K respectively
3       17304
4       20462   32750   4K & 16K respectively
```

To turn cursor OFF poke 0 into the address.
To turn it back on poke 208 into the address.

Any value may be POKEd in to turn the cursor back on and this will give a different character.

　　　　　　　　　　　　　　　- 0000000000 -

　　　　　　　　　　　　***** INPUT/OUTPUT *****

From David Harris, 470 Marion Road, Plympton Park, South Australia 5038
I am interested in developing the use of my TRS-80 for computer-aided design (C.A.D). I am
an engineer and have already developed some programs on the fringe of C.A.D. but am looking
to take it further. I would like to contact anyone interested in this type of application and
request that any interested reader contact me on 'phone number (08) 297 8207 during office hours.

(It sounds an interesting and challenging application, David. Would any interested readers
please contact David directly - Ed.)

From: Mr. R. Hutson - London SE 12.

I would like to comment on the letter from B.N. Hall on page 11 of issue 19, where cassette waveform problems were described and a method of reproducing the correct waveform was included. This method required two comparators and two timers; these used the asymmetrical waveform to generate symmetrical pulses.

I believe that I have an answer to the problem using a simple single stage transistor circuit. I approached the problem by reproducing a pulse of similar shape and frequency from a pulse generator, from this I received a small positive response and a large negative response from my CTR41 cassette recorder. Upon further experimentation, I found that by sending the negative part of the pulse first (inversion), the waveform was reproduced perfectly, as the TRS-80 rectifies the audio input for processing, and inverting the pulse makes no difference.

Inversion was achieved by a single stage transistor circuit (audio amplifier), shown below with excellent results.

I have not investigated further as the problem had been solved, but I have had no dropouts since. I would be interested in a full explanation of why the recorder should prefer negative pulse before working.   I would also be interested in a high speed cassette which I will start work on in the near future (approx. 4800 band).
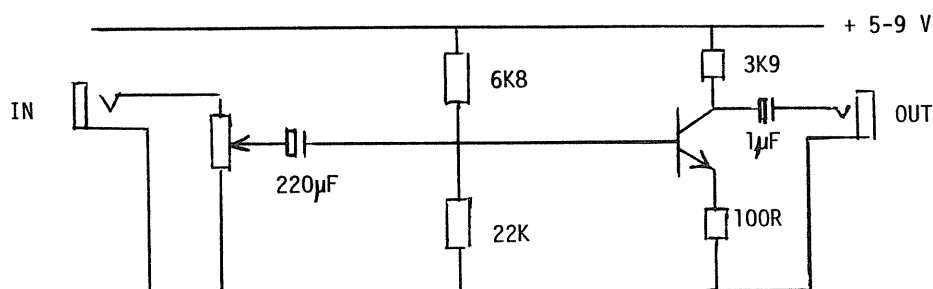
(Thank you for this contribution Mr. Hutson.   We have recently been carrying out an investigation into the cassette interface with a view to improving the quality of the cassettes we produce. We observed the phenomenon you describe but, as yet, have no explanation for it.   It not only occurs in the relatively unsophisticated CTR41/CTR80/Video Genie cassette recorders but also in much more expensive cassette decks and high speed amplifiers.   I suspect the real answer lies in the fact that we are dealing with a very non-linear system.   Although the waveform of each pulse approaches a sinusoid after passing through the amplifier and being differentiated by the recording process, it should be remembered that each pulse is a single event and steady state theory does not apply - at least, not to the pulses as we see them on an oscilloscope. I seem to vaguely remember in my student days using Fourier Transformers to express signals such as these in steady-state terms.   Perhaps one of our readers who is more au fait with these matters could offer an explanation (but please do not make it too mathematical).   On the other hand, there may be a simple explanation - we would love to hear it.   Concerning the high speed cassette interface you mention - one at least has already been developed in the U.S.A.   It is called the "Poor Man's Floppy" model TC-8.   It is available from JPC PRODUCTS CO. 12021 Paisano Ct. Albuquerque, New Mexico 87112 and operates at 2500 band.   It sells in kit form for US$90 and assembled for US$120.   We understand that in Australia De Forest Software, Station Street, Nunawading, Vic., also has these units available - Ed.)

From: Murray J. Dixon - Ringwood Victoria.

I would like to comment further on the letter from your UK correspondent (November 1980) regarding educational uses of the '80.   Many people might be surprised to find the extent to which micro-computers are being used in secondary schools throughout Australia (our own small region has approximately 50 TRS-80 and System 80 computers in about half a dozen schools).   This being the case, it is quite possible that a significant number of your readers are involved in computer education of some form.   Could this readership be sufficiently large for you to consider including in each issue, a program which has a formal educational purpose as its main function.

Your comments regarding educational use of computers in schools in the UK are quite valid so, by pursuing such a course, perhaps you could assist we "beginners" in Australia by encouraging contributions from your more experienced UK subscribers.

(We would be only too happy to publish more educational programs and related material, Murray, if only it were forthcoming.   Unfortunately, many of the "educational" programs submitted to us for consideration are no more than testing programs to ascertain how much a child knows. Whilst such programs have their place, there is a limit to the number of such programs which can be usefully published.   It is strange, because we know that a significant factor for many people when deciding to purchase a computer is its educational value to their children.   We will certainly be happy to publish as many good educational programs as possible and will also be happy to do our bit in helping to disseminate information about this field.   In fact, we have some interesting articles on computers in education in the pipeline already.   You might also note the '80 Users' Groups this month - some educationally oriented groups are now listed and I am sure they would welcome participation from any of our readers. - Ed.)

MORE AUSTRALIAN SOFTWARE

All programs designed to run on both the TRS-80 or the SYSTEM 80 without modification. Most programs include sound

TRIAD VOL 1 — L2/16K
Cassette $10.95 Disk $15.95
+ 60c p&p

Three separate games which test your powers of memory and concentration. The programs combine graphic displays and sound:

SIMON-SEZ: Just like the electronic music puzzles on sale for more than $20. Numbers are flashed on the screen and sounded in a sequence determined by the computer. Your task is to reproduce the sequence, correctly.

LINE?: Rather like a super, complicated version of noughts and crosses. You may play against another player or against the computer itself. But beware, the computer cheats!

SUPER CONCENTRATION: Just like the card game but with more options. You must find the hidden pairs. You may play against other people, play against the computer, play on your own, or even let the '80 play on its own.

TRIAD VOL 2 — L2/16K
Cassette $10.95 Disk $15.95
+ 60c p&p

Remember those "NUMERO" puzzles in which you had a matrix of numbers (or letters) with one blank space and you had to shuffle the numbers around one at a time until you had made a particular pattern? Well, **SHUFFLEBOARD**, the first program in this triad, is just this, except that the computer counts the number of moves you take to match the pattern it has generated — so it is not possible to cheat.

MIMIC is just like SHUFFLEBOARD except that you only see the computer's pattern for a brief span at the beginning of the game, then you must remember it!

In **MATCHEM**, you have to manoeuvre 20 pegs from the centre of the screen to their respective holes in the top or bottom rows. Your score is determined by the time taken to select a peg, the route taken from the centre of the screen to the hole and your ability to direct the peg into the hole without hitting any other peg or the boundary.

VISURAMA L2/16K
Cassette $10.95 Disk $15.95
+ 60c p&p

Two programs which give fascinating, ever-changing patterns on the screen.

LIFE is the fastest implementation of the Game of Life you will see on your '80. Machine language routines create up to 1200 new generations per minute for small patterns or up to 100 per minute for the full 128 x 48 screen matrix. Features full horizontal and vertical wraparound.

EPICYCLES will fascinate you for hours. The ever-changing ever-moving patterns give a 3D effect and were inspired by the ancient Greek theories of Ptolemy and his model of the Solar system.

EDUCATION AND FUN — L1/4K, L2/16K
Cassette $10.95 Disk $15.95
+ 60c p&p

Written by a primary school teacher to make learning enjoyable for his pupils, there are five programs in both Level I and Level II to suit all systems:

BUG-A-LUG: a mathematics game, in which you must get the sum correct before you can move.

AUSTRALIAN GEOGRAPHY: learn about Australian States and towns, etc.

SUBTRACTION GAME: build a tower with correct answers.

HOW GOOD IS YOUR MATHS? Select the function (+, −, ÷ or X) and degree of difficulty.

HANGMAN: That well known word game now on your computer.

Recommended for children from 6 to 9 years.

COSMIC FIGHTER & SPACE JUNK — L2/16K
Cassette $10.95 Disk $15.95
+ 60c p&p

Both programs have sound to complement their excellent graphics. In **COSMIC FIGHTER**, you must defend the earth against seven different types of alien aircraft. It is unlikely that you will be successful but you will have a lot of fun trying!

You mission in **SPACE JUNK** is to clean up all the debris left floating around in space by those other space games. It is not as simple as it sounds and space junk can be quite dangerous unless you are very careful.

SPACE DRIVE L2/4K & 16K
Cassette $8.95 Disk $13.95
+ 60c p&p

Try to manoeuvre your space ship through the meteor storms then land it carefully at the space port without running out of fuel or crashing. Complete with realistic graphics.

STARFIRE AND NOVA INVASION L2/16K
Cassette $10.95 Disk $15.95
+ 60c p&p

Both programs include sound to improve their realism.

STARFIRE seats you in the cockpit of an X-wing fighter as you engage in battle with the deadly Darth Vader's Tie-fighters. Beware of the evil one himself and may the Force be with you.

In **NOVA INVASION**, you must protect your home planet of Hiberna from the invading NOVADIANS. You have two fixed guns at each side of the screen and a moveable one at the bottom. Apart from shooting down as many invaders as possible, you must protect your precious hoard of Vitaminium or perish!

AIR ATTACK AND NAG RACE — L2/16K
Cassette $10.95 Disk $15.95
+ 60c p&p

An unlikely combination of programs but they share the same author who has a keen sense of humour.

AIR ATTACK includes sound and realistic graphics. The aircraft even have rotating propellors! But they also drop bombs on you, so it's kill or be killed!

NAG RACE lets you pander to your gambling instinct without actually losing real money. Up to five punters can join in the fun. Each race results in a photo-finish whilst there is a visible race commentary at the bottom of the screen throughout the race. Happy punting!

FOUR LETTER MASTERMIND L2/16K
Cassette $8.95 Disk $13.95
+ 60c p&p

There are 550 four-letter words from which the computer can make its choice. You have 12 chances to enter the correct word. After each try, the computer informs you of the number of correct letters and those in the correct position. You can peek at the list of possible words but it will cost you points. Makes learning to spell fun.

MUSIC IV — L2/16K
Cassette $8.95 Disk $13.95
+ 60c p&p

Music IV is a music compiler for your '80. It allows you to compose or reproduce music with your computer that will surprise you with its range and quality. You have control over duration (full beat to 1/16 beat) with modifications to extend the duration by half or one third for triplets. Both sharps and flats are catered for as are rests. Notes on whole sections may be repeated. The program comes with sample data for a well-known tune to illustrate how it is done.

1.4 MEGABYTES ON LINE + 48K RAM
for $3800 incl. Sales Tax

MICRO-80's

MODEL 380 +

MICRO-80 has equipped the TRS-80 with two high reliability dual-head 80 track mini-floppy disk drives made by MPI, one of America's leading mini-disk drive manufacturers.

This turns the mild-mannered Model 3 into a powerhouse able to handle the most difficult business programs. The TRS-80 is one of the best-supported microcomputers in the world. MICRO-80 has been supporting the TRS-80 in Australia for 18 months and is one of Australia's leading dealers in MPI disk drives.

2.8 MEGABYTES FOR $5300 incl. Sales Tax

If you need even more file space you can add MICRO-80's external dual-drive cabinet enclosing two more dual-head 80 track drives for an additional $1500.

COMPUTER PRICES

MODEL 340
2 40 TRACK SINGLE HEAD DRIVES GIVING
350K FORMATTED STORAGE, 48K RAM

$2990 INCL. SALES TAX

MODEL 340 +
2 40 TRACK DUAL-HEAD DRIVES GIVING
700K FORMATTED STORAGE, 48K RAM

$3350 INCL. SALES TAX

MODEL 380
2 80 TRACK SINGLE HEAD DRIVES GIVING
700K FORMATTED STORAGE, 48K RAM

$3350 INCL. SALES TAX

MODEL 380 +
2 80 TRACK DUAL-HEAD DRIVES GIVING
1.4 MEGABYTE FORMATTED STORAGE, 48K RAM

$3800 INCL. SALES TAX

350K SYSTEM
MODEL 340, EPSON MX-80 PRINTER
NEWDOS 80 DISK OPERATING SYSTEM

$4070 INCL. SALES TAX

700K SYSTEM (40 Track)
MODEL 340 +, EPSON MX-80 PRINTER
NEWDOS 80 DISK OPERATING SYSTEM

$4429 INCL. SALES TAX

700K SYSTEM (80 Track)
MODEL 380, EPSON MX-80 PRINTER
NEWDOS 80 DISK OPERATING SYSTEM

$4429 INCL. SALES TAX

1.4 MEGABYTE SYSTEM
MODEL 380 +, EPSON MX-80 PRINTER
NEWDOS 80 OPERATING SYSTEM

$4880 INCL. SALES TAX

2.8 MEGABYTE SYSTEM
MODEL 380 +, DUAL EXTERNAL DRIVES,
MX-80 PRINTER, NEWDOS 80 OPERATING SYSTEM

$6380 INCL. SALES TAX

★

EXATRON STRINGY FLOPPY — $372.50 Incl. P&P

All Exatron Stringy Floppies sold by MICRO-80 include the special chained version of **HOUSEHOLD ACCOUNTS,** developed by Charlie Bartlett. When used on the ESF, this program is powerful enough to perform many of the accounting functions in a small business. Remember, the ESF comes complete with a comprehensive manual, a 2 way bus-extender cable, its own power supply and 10 wafers of mixed length. One wafer contains the Data Input/Output program and another the **HOUSEHOLD ACCOUNTS** program.

CAN'T MAKE UP YOUR MIND ABOUT THE ESF?

Then send in $5.00 for a copy of the manual. We will refund your $5.00 IN FULL when you purchase an ESF.

★

AUSTRALIAN SOFTWARE (Cont.)

GAMES

SHEEPDOG (L2/16K) $8.95 + 60c p&p
Ever wondered how a sheepdog manages to drive all those awkward sheep into a pen? Well, here is your chance to find out just how difficult it is and have a lot of fun at the same time. You control the sheepdog, the computer controls the sheep! As if that isn't enough, look out for the dingoes lurking in the bush!

U BOAT $8.95 + 60c p&p
Real time simulation at its best! Comes with working sonar-screen and periscope, a full rack of torpedoes, plenty of targets, working fuel and battery meters, helpful Mothership for high-seas reprovisioning and even has emergency radio for that terrible moment when the depth charges put your crew at risk. Requires Level II/16K.

SPACE INVADERS WITH SOUND $8.95 + 60c p&p
Much improved version of this arcade favourite with redesigned laser and cannon blasts, high-speed cannon, 50 roving drone targets, 10 motherships and heaps of fun for all. Level II with 4K and 16K versions on this cassette.

GOLF (L2/16K) $8.95 + 60c p&p
Pit your skills of mini-golf against the computer. Choose the level of difficulty, the number of holes and whether you want to play straight mini golf or crazy golf. Complete with hazards, water traps, bunkers and trees. Great fun for kids of all ages.

DOMINOES(L2/16K) $8.95 + 60c p&p
Pit your skill at dominoes against the computer, which provides a tireless opponent. Another application of supergraphics from the stable of Charlie Bartlett. Dominoes are shown approximately life size in full detail (except for colour!). The monitor screen is a window which you can move from one end of the string of dominoes to the other. Best of all, you don't lose any pieces between games!

KID'S STUFF (formerly MMM-1) $8.95 + 60c p&p
Three games on one cassette from that master of TRS-80 graphics, Charlie Bartlett. Includes INDY 500, an exciting road race that gets faster and faster the longer you play, SUBHUNT in which your warship blows up unfortunate little submarines all over the place, and KNIEVEL (as in motorcycle, ramp and buses).

OTHER PROGRAMS

INFINITE BASIC BY RACET (32K/1 DISK)
$49.95 + 50c. p&p
Full matrix functions — 30 BASIC commands; 50 more STRING functions as BASIC commands.

GSF/L2/48K $24.95 + 50c. p&p
18 machine language routines including RACET sorts.

BUSINESS ADDRESS AND INFORMATION SYSTEM
(48K/DISK) $24.95 + 50c. p&p
Allows you to store addresses and information about businesses, edit them and print them out.

HISPED (L2 16, 32 or 48K) $29.95
This machine language program allows you to SAVE and LOAD programs and data to tape at speeds up to 2000 band (4 times normal) using a standard cassette recorder. A switch must be installed to remove the XRX III loading board, if fitted.

LOWER CASE FOR YOUR TRS-80/SYSTEM 80
Kit only $49.00 plus $2.00 p&p
Give your TRS-80 or SYSTEM 80 a lower case display with proper descenders and a block cursor (similar to the TRS-80 Model III). Also includes symbols for the four suits of cards. Includes full fitting instructions, all necessary components and a special machine language driver program to enable lower case in BASIC. The modification is similar to the Tandy model and does not work with Electric Pencil without further modifications.

These kits require disassembly of your computer and some soldering. They should only be installed by someone who has experience in soldering integrated circuits, using a low power, properly earthed soldering iron. If you do not have the necessary experience/ equipment, we will install the modification for you for $20 plus freight in both directions. Make sure you arrange the installation with us first, before despatching your computer, so that we can assure you of a rapid turn-around. We are also arranging to have installers in each State. See elsewhere in this issue for their names and addresses.

PRICES
Cat No.
HD-020 Lower case mod kit for TRS-80
$49.00 plus $2.00 p&p
HD-021 Lower case mod kit for SYSTEM-80
$49.00 plus $2.00 p&p

EPSON MX-80 PRINTER
ONLY *$949 Inc. Cable for TRS-80 and p&p
(*Printer only — $940 incl. p&p)
The EPSON MX-80 printer is compact, quiet, has features unheard of only 2-3 years ago in a printer at any price and, above all, is ultra-reliable. All available print modes may be selected under software control. Features include:
- high quality 9x9 dot-matrix character formation
- 3 character densities
  . 80 characters per line at 10 chars/inch
  .132 characters per line at 16.5 chars/inch
  . 40 characters per line at 5 chars/inch
- 2 line spacings
  . 6 lines per inch   8 lines per inch
- 80 characters per second print speed
- bi-directional printing
- logical seeking of shortest path for printing
- lower case with descenders
- TRS-80 graphics characters built in
- standard Centronics printer port

The bi-directional printing coupled with the logical seeking of the shortest print path (which means that the print head will commence printing the next line from the end which requires the least travel, thereby minimising unutilised time) gives this printer a much higher throughput rate than many other printers quoting print speeds of 120 c.p.s. or even higher.

GREEN SCREEN SIMULATOR
$9.50 incl. p&p
The GREEN SCREEN SIMULATOR is made from a deep green perspex, cut to fit your monitor. It improves contrast and is much more restful to the eyes than the normal grey and white image.

All editorial staff of MICRO-80 are now using GREEN SCREEN SIMULATORS on their own monitors.

Please make sure to specify whether you have an old (squarish) or new (rounded) style monitor when ordering. Not available for Dick Smith monitors.

***** PUT   m/l 4-16K   by S. Barnett *****

This is a m/l program to SET nongraphic characters. You will need either a monitor program such as TBUG or BMON (MICRO-80 issues 3, 4 and 5) or an editor/assembler. If using a monitor, start editing memory at the address shown at the top of the left hand column and change each memory address to the value shown in the next column. E.g. change 7FB8 to ,   to , to ,   to ,   to   and so on. Note that this program resides in the same area as BMON. If you wish to use BMON to enter it, then you must either use the 32K or 48K version (assuming your computer has sufficient capacity) or you must offset the program to start at 5FB8H (say) and append a short block move routine as described in MICRO-80 issue 10 to shift it to its correct position when run. Using the Editor/Assembler, you may easily relocate the program for different size machines simply by changing the ORG address.   Use 4FB8 for   4K, BFB8 for 32K and FFB8 for 48K machines.

Use the editor/assembler or monitor to punch out a SYSTEM tape having the following parameters:

START   END   ENTRY   NAME
7FB8   PUT

Note, if you have appended a block move routine, the START, END and ENTRY addresses will differ from those shown above.

To load the program from tape type:-

SYSTEM and press the ENTER/NEWLINE key. Then in response to the next prompt type:-

PUT and press the ENTER/NEWLINE key. When the program has loaded type / and press the ENTER/ NEWLINE key. PUT is now active.

Although this program was not designed to RUN on a disk system it will do so without any modification. Of course, once the program has been loaded, disk users should be aware that the normal function of the PUT command will now not be available until the system is reBOOTed.

To load the program from disk, if you have NEWDOS 80, type:-

BASIC press ENTER/NEWLINE
CMD"PUT" press ENTER/NEWLINE

For any other DOS:-

From DOS type LOAD PUT/CMD and press ENTER/NEWLINE
Type BASIC and press ENTER/NEWLINE
Type SYSTEM and press ENTER/NEWLINE
Type /32696 and press ENTER/NEWLINE

DO NOT attempt to start the program from DOS or your system will hang.

You may NOW PUT non graphics on the screen in a similar fashion to the SET command.
The format of the command can be either of the following:-

PUT(X,Y), "stringname"   or   PUT(X,Y),variable

X can be any value between 0 and 63 corresponding to its position across the screen and Y can be any value between 0 and 15 corresponding to its position down the screen.

For example, PUT(20,15),"*" will SET an asterisk on the screen 20 columns across and 15 lines down; the same thing may be done this way:-

```
10  CLS
20  A$="*"
30  PUT(20,15),A$
```

The normal SN ERROR will be given if the command is not typed in correctly or an FC ERROR will be given if you specify values for X or Y that exceed the stated range.

```
                       00001 ;      (C) S. BARNETT
                       00002 ;      PUT COMMAND TO SET NON GRAPHICS
      7FB8             00003       ORG      7FB8H
      7FB8  21C67F     00004 START LD       HL,GO     ;INSTALL
      7FBB  228341     00005       LD       (4183H),HL    ;PUT
      7FBE  3EC3       00006       LD       A,0C3H        ;COMMAND
      7FC0  328241     00007       LD       (4182H),A
      7FC3  C3CC06     00008       JP       6CCH          ;RETURN TO BASIC
      7FC6  CF         00009 GO    RST      08H           ;SN ERROR IF NOT   "("
      7FC7  28         00010       DEFB     28H
      7FC8  CD1C2B     00011       CALL     2B1CH         ;EVALUATE X PARAMETER
      7FCB  FE40       00012       CP       64            ;IS IT GREATER THAN OR
      7FCD  D24A1E     00013       JP       NC,1E4AH  ;EQUAL TO 64. IF YES FC ERROR
      7FD0  F5         00014       PUSH     AF            ;SAVE X PARAMETER
      7FD1  CF         00015       RST      08H           ;SN ERROR IF NOT ","
```

```
7FD2 2C        00016        DEFB   2CH
7FD3 CD1C2B    00017        CALL   2B1CH        ;EVALUATE Y PARAMETER
7FD6 FE10      00018        CP     16           ;FC ERROR IF Y VALUE IS
7FD8 D24A1E    00019        JP     NC,1E4AH     ;GREATER OR EQUAL TO 16
7FDB 22FD7F    00020        LD     (TMPS),HL    ;SAVE COMMAND POINTER
7FDE 3C        00021 XX1    INC    A            ;MAKE SURE Y PARAMETER IS
               00022 ;                          AT LEAST 1
7FDF 47        00023        LD     B,A          ;LOAD IN B FOR LOOP
7FE0 210000    00024        LD     HL,00        ;CLEAR HL
7FE3 114000    00025        LD     DE,64        ;DE=LENGTH OF FULL LINE
7FE6 19        00026 LOOP1  ADD    HL,DE        ;LOOP DOWN SCREEN
7FE7 10FD      00027        DJNZ   LOOP1
7FE9 ED52      00028        SBC    HL,DE        ;ADJUST FOR ADDITIONAL
               00029 ;                          ;COUNT AT XX1
7FEB F1        00030        POP    AF           ;RESTORE X VALUE
7FEC 163C      00031        LD     D,3CH        ;DE=POSITION ACROSS
7FEE 5F        00032        LD     E,A          ;SCREEN.
7FEF 19        00033        ADD    HL,DE        ;ADD DISPLACEMENT DOWN
               00034 ;                          ;SCREEN.
7FF0 222040    00035        LD     (4020H),HL   ;SAVE IN CURSOR POINTER
               00036 ;                          ;ADDRESS.
7FF3 2AFD7F    00037        LD     HL,(TMPS)    ;RESTORE COMMAND POINTER
7FF6 CF        00038        RST    08H          ;SN ERROR IF NOT ")"
7FF7 29        00039        DEFB   29H
7FF8 CF        00040        RST    08H          ;SN ERROR IF NOT ","
7FF9 2C        00041        DEFB   2CH
7FFA C3AD20    00042        JP     20ADH        ;JUMP TO PRINT ROUTINE
7FFD 0000      00043 TMPS   DEFW   0H           ;WORKSPACE
7FB8          00044        END    7FB8H
00000 TOTAL ERRORS
```

***** SHORTEN 32K m/l DISK   by Spencer George *****

This is a machine language program for a 32K disk system, it may be modified for a 16K or 48K machine by altering the ORG statement to 7400H for 16K FE00H for 48K. The version listed loads from BE00 to BEC8.

The program SHORTENs basic programs by removing unnecessary blanks, linefeeds and REMark statements. The program is fairly simple-minded in its operation, i.e. no checks are made to see if the REM statement being removed is a GOTO or GOSUB branch line, therefore you should ALWAYS make sure that you SAVE a copy of the program to disk first, which is not only good sense, but will be handy for cross checking later in case any REM's were a GOTO or GOSUB destination. Better still, don't make REM's the destination for a GOTO or GOSUB in the first place, it's not good programming practise. Having thus cautioned you against what it doesn't do, what it does do is very useful and very quick. To use SHORTEN follow these very simple steps:-

1. Go into BASIC and .... LOAD"program/BAS"

2. Save your program back to disk...... SAVE"oldname/BAS"

3. Jump back to DOS by pressing reset or type ... CMD"S"

4. Start the m/l program by typing SHORTEN or SHORTEN/CMD

5. Jump back to the altered program by typing....BASIC * so that the pointers are not reset.

6. Save the program back to disk... SAVE"newname/BAS"

Alright, the smarty at the back of the class just typed ?MEM and got the same answer back as he did before the program was SHORTENed, but look!! All the REM's and blanks have gone - why the same answer to ?MEM. Simple...we returned to BASIC from DOS with the command BASIC* which does not reset the pointers so that our program will still be there and the pointers are still set to the value of the old version. To see how much memory you have saved try this.....

1. Type ?MEM or PRINT MEM and make a note of the number.

2. Provided you saved the program to disk as SAVE"newname/BAS" then return to DOS as in STEP 3 above.

3. Return to BASIC, this time resetting the pointers, by typing BASIC.

4. Type LOAD"newname/BAS"

5. Now type ?MEM or PRINT MEM and compare it to the value obtained in step 1 to see how much memory has been saved.

When you are satisfied that your program is still functioning correctly, i.e. no UL ERROR messages, then return to DOS and type KILL oldname/BAS then RENAME newname/BAS oldname/BAS

ENTERING THE PROGRAM

You may enter the program using the M - command in DEBUG or by using a disk-based editor/assembler.
If you use DEBUG you will need to use the TRSDOS DUMP command to transfer the program from memory
to disk with the following parameter:

	FILESPEC	START	END	ENTRY
	SHORTEN/CMD	BE00	BEC8	BE00

```
BE00              00001         ORG     0BE00H
BE00  D9          00002 AAN     EXX
BE01  0605        00003         LD      B,AAD
BE03  D9          00004 AAO     EXX
BE04  08          00005         EX      AF,AF'
BE05  3E00        00006         LD      A,AAA
BE07  08          00007         EX      AF,AF'
BE08  DD21A440    00008         LD      IX,AAM
BE0C  DD5E00      00009         LD      E,(IX+00H)
BE0F  DD5601      00010         LD      D,(IX+01H)
BE12  DD6E00      00011         LD      L,(IX+00H)
BE15  DD6601      00012         LD      H,(IX+01H)
BE18  010400      00013         LD      BC,AAC
BE1B  EDB0        00014         LDIR
BE1D  08          00015 AAP     EX      AF,AF'
BE1E  3C          00016         INC     A
BE1F  08          00017         EX      AF,AF'
BE20  7E          00018         LD      A,(HL)
BE21  FE00        00019         CP      AAA
BE23  281D        00020         JR      Z,AAR
BE25  FE22        00021         CP      AAH
BE27  283A        00022         JR      Z,AAU
BE29  FE93        00023         CP      AAK
BE2B  2849        00024         JR      Z,AAX
BE2D  FE27        00025         CP      AAI
BE2F  CA76BE      00026         JP      Z,AAX
BE32  FE0A        00027         CP      AAF
BE34  2876        00028         JR      Z,ABE
BE36  FE20        00029         CP      AAG
BE38  2826        00030         JR      Z,AAT
BE3A  FE3A        00031         CP      AAJ
BE3C  2872        00032         JR      Z,ABF
BE3E  EDA0        00033 AAQ     LDI
BE40  18DB        00034         JR      AAP
BE42  EDA0        00035 AAR     LDI
BE44  EDA0        00036         LDI
BE46  7E          00037         LD      A,(HL)
BE47  FE00        00038         CP      AAA
BE49  280B        00039         JR      Z,AAS
BE4B  010300      00040         LD      BC,AAB
BE4E  EDB0        00041         LDIR
BE50  08          00042         EX      AF,AF'
BE51  3E00        00043         LD      A,AAA
BE53  08          00044         EX      AF,AF'
BE54  18C7        00045         JR      AAP
BE56  EDA0        00046 AAS     LDI
BE58  CDF81A      00047         CALL    AAL
BE5B  D9          00048         EXX
BE5C  10A5        00049         DJNZ    AAO
BE5E  D9          00050         EXX
BE5F  C9          00051         RET
BE60  23          00052 AAT     INC     HL
BE61  18BA        00053         JR      AAP
BE63  EDA0        00054 AAU     LDI
BE65  7E          00055 AAV     LD      A,(HL)
BE66  FE00        00056         CP      AAA
BE68  28D8        00057         JR      Z,AAR
BE6A  FE22        00058         CP      AAH
BE6C  2804        00059         JR      Z,AAW
BE6E  EDA0        00060         LDI
BE70  18F3        00061         JR      AAV
BE72  EDA0        00062 AAW     LDI
BE74  18A7        00063         JR      AAP
BE76  08          00064 AAX     EX      AF,AF'
BE77  FE07        00065         CP      AAE
BE79  F28FBE      00066         JP      P,ABB
BE7C  08          00067         EX      AF,AF'
```

```
BE7D  1B          00068            DEC    DE
BE7E  1A          00069            LD     A,(DE)
BE7F  FE3A        00070            CP     AAJ
BE81  2001        00071            JR     NZ,AAY
BE83  1B          00072            DEC    DE
BE84  1B          00073 AAY        DEC    DE
BE85  1B          00074            DEC    DE
BE86  1B          00075 AAZ        DEC    DE
BE87  23          00076 ABA        INC    HL
BE88  7E          00077            LD     A,(HL)
BE89  FE00        00078            CP     AAA
BE8B  2805        00079            JR     Z,ABC
BE8D  18F8        00080            JR     ABA
BE8F  08          00081 ABB        EX     AF,AF'
BE90  18F4        00082            JR     AAZ
BE92  08          00083 ABC        EX     AF,AF'
BE93  FE07        00084            CP     AAE
BE95  F2A9BE      00085            JP     P,ABD
BE98  08          00086            EX     AF,AF'
BE99  23          00087            INC    HL
BE9A  EDA0        00088            LDI
BE9C  7E          00089            LD     A,(HL)
BE9D  FE00        00090            CP     AAA
BE9F  28B5        00091            JR     Z,AAS
BEA1  010300      00092            LD     BC,AAB
BEA4  EDB0        00093            LDIR
BEA6  C31DBE      00094            JP     AAP
BEA9  08          00095 ABD        EX     AF,AF'
BEAA  1896        00096            JR     AAR
BEAC  23          00097 ABE        INC    HL
BEAD  C31DBE      00098            JP     AAP
BEB0  23          00099 ABF        INC    HL
BEB1  7E          00100            LD     A,(HL)
BEB2  FE00        00101            CP     AAA
BEB4  288C        00102            JR     Z,AAR
BEB6  2B          00103            DEC    HL
BEB7  1885        00104            JR     AAQ
BEB9  08          00105            EX     AF,AF'
BEBA  FE07        00106            CP     AAE
BEBC  F28FBE      00107            JP     P,ABB
BEBF  08          00108            EX     AF,AF'
BEC0  1B          00109            DEC    DE
BEC1  1B          00110            DEC    DE
BEC2  1B          00111            DEC    DE
BEC3  1B          00112            DEC    DE
BEC4  18C0        00113            JR     AAZ
0000              00114 AAA        EQU    0000H
0003              00115 AAB        EQU    0003H
0004              00116 AAC        EQU    0004H
0005              00117 AAD        EQU    0005H
0007              00118 AAE        EQU    0007H
000A              00119 AAF        EQU    000AH
0020              00120 AAG        EQU    0020H
0022              00121 AAH        EQU    0022H
0027              00122 AAI        EQU    0027H
003A              00123 AAJ        EQU    003AH
0093              00124 AAK        EQU    0093H
1AF8              00125 AAL        EQU    1AF8H
40A4              00126 AAM        EQU    40A4H
BE00              00127            END    AAN
00000 TOTAL ERRORS
```

***** POKER L2/16K   by S. Franzon *****

When the game starts five cards are displayed on the screen face down, the program asks you to BET by pressing the "B" key.   One credit is subtracted from your total credit of 10 each time that you press "B".   When you have bet the desired amount press "D" to draw, the cards will then be turned over.   You should now decide which cards to keep. To keep a card, type the number that appears above each card, the word HELD will appear over the respective card.   To cancel HELD cards press "C".   When you have HELD the cards that you wish to keep press "D" again and any card not HELD will be replaced with another one and displayed on the screen.   Program logic will then decide what if anything you have won.   Winning hands are based on the normal rules for POKER. The game continues until you run out of credits.

```
BETS =  1                                    CREDITS =  8
        PAYOUTS
TWO PAIR = 2     FULL HOUSE = 10      PRESS
3 / KIND = 3     4 / KIND   = 40      1 - 5 TO HOLD CARDS
STRAIGHT = 5     STR/FLUSH  = 100     C TO CANCEL HELD CARDS
FLUSH    = 7     RYL/FLUSH  = 400     D TO DRAW
        HELD         HELD        3         HELD         HELD
```

```
    NINE        NINE                    SEVEN       SEVEN


      ♥           ♦         (card)        ♦           ♠


         NINE        NINE                    SEVEN       SEVEN
```
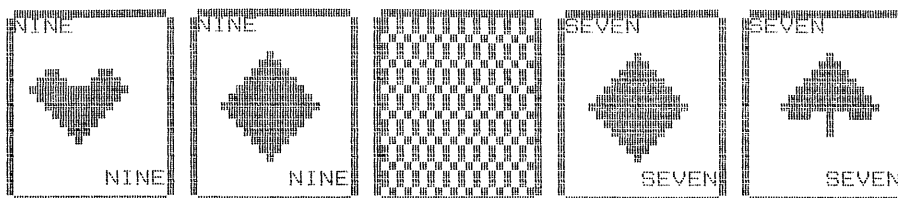
```
BETS =  3                                    CREDITS =  4
        PAYOUTS
TWO PAIR = 2     FULL HOUSE = 10      PRESS
3 / KIND = 3     4 / KIND   = 40      1 - 5 TO HOLD CARDS
STRAIGHT = 5     STR/FLUSH  = 100     C TO CANCEL HELD CARDS
FLUSH    = 7     RYL/FLUSH  = 400     D TO DRAW
        HELD         HELD        3         4           5
```

```
    NINE        NINE        TWO         KING        TEN


      ♦           ♣          ♠           ♣           ♥


         NINE        NINE        TWO        KING         TEN
```
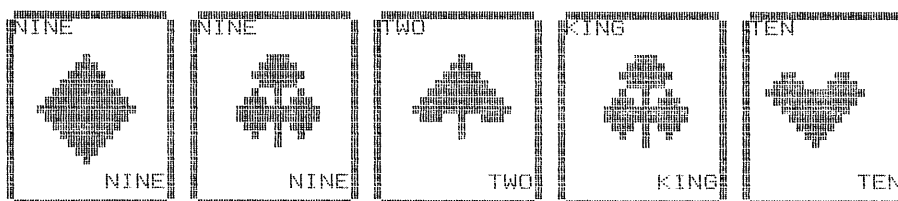
The program structure and variables are explained below.

100-130	Initialization.
140-210	Screen set up-printing payouts and drawing cards.
220-350	Randomly choosing cards, and check that each card isn't used twice.
360-430	INKEY$ to bet and to deal cards.
Subs 450, 460, 470, 480, 490	are where the starting points for the graphics of each card are kept.
500-640	INKEY$ to hold cards and to redeal the cards.
650-720	Replace the cards not held.
730-780	Put the number values of each card (i.e. 1 for Ace, 2 for two...) into increasing order for checking wins.
790-840	Check for winning branches to subroutines
850	Check for 4/kind
940	Check for 3/kind then full house.
940	Check for pairs.
1010	Check for flush then GOSUB 1050 to check for royal or straight flush.
1050	Check for straight.
	After branching to each subroutine it checks if there was a win and it then will branch away if it is so.
1080-1180	Prints out any winnings.
1190-1250	INKEY$ for double up or collect winnings.
1260-1300	Print out double payments and draws card.
1310-1350	INKEY$ for choosing big or small.
1360-1480	Checks for win or lose in double up and draws graphics on cards.
1490	Checks if you have any credits left.
1520-1540	Used when no win occurs.
Sub 1550-1610	Draws outline of cards.
Sub 1620-1640	Draws backing on cards and clears away backing.
Sub 1650-1710	Draws suits on cards.
Sub 1720-1890	Four different subs for setting graphics of each suit.
Sub 1900-1950	Writes on the value of each card (i.e. Ace).
Sub 1960-2010	Instructions.

The subroutines between 450-490 are used must commonly as they are called every time a change takes place to a card. The most important variables used are -

A$(1-13)   Written value of each card i.e. A$(1)="ACE". A$(2)="two".
A(0-4)   Hold the value for each of the card Nos. (i.e. 1 to 13)
           N.B. A(1) is card one, A(2) card two, etc.
D(0-4)   Holds the suit value (i.e. 1 to 4)
B(0-4)   Holds   the value of A(0-4)+B(0-4)x20 (i.e. B(0)=A(0)+B(0)x20).   This means every one
           of the 52 cards in the deck has a different B(0-4) value. Each value is checked against
           each other so no other card is the same.
C(0-4)   Holds the B(0-4) value when that card is being replaced (i.e. B(0-4) will get a new
           value and the old value is kept in C(0-4) and is checked against the new value so
           the same card can't be drawn after it was discarded.
P(0-7)   Hold the values of the payouts.
V(0-4)   Gets a value of one when it is held so the computer can tell which cards have been
           held.
C   holds the credit value
M   holds the bet value for each hand
E   Is the bet (M) x payout when a win takes place.   It is equal to the amount won (i.e.
           bet 2, won 2 pair = 2 x 2 = 4)
V   Holds the value of the winnings (i.e. gets a different value for each different type
           of win)
F   Equals 165 when GOSUB 1620 to draw backing on card and equals 128 when used to clear
           backing.
A,T,B,Y,Z,H   are starting points for the outline of the card, the backing of the card and for
           drawing the graphics of the suit.

If you wish to start with more credits, change C (credits) in Line 40.

If you wish the payouts to be higher, change P(0-7) (payouts) in Line 120.

```
40 RANDOM:C=10:DIMA$(13)
50 CLS:PRINT@405,CHR$(23);"P O K E R ";
60 PRINT@512,"DO YOU WANT INSTRUCTIONS <Y/N>";
70 A$=INKEY$:IFA$=""GOTO70
80 IFA$="Y"GOSUB1960:GOTO100
90 IFA$<>"N"GOTO70
100 FORQ=1TO13:READA$(Q):NEXTQ
110 DATA"ACE","TWO","THREE","FOUR","FIVE","SIX","SEVEN","EIGHT",
"NINE","TEN","JACK","QUEEN","KING"
120 P(0)=2:P(1)=3:P(2)=5:P(3)=7:P(4)=10:P(5)=40:P(6)=100:P(7)=40
0
130 FORQ=0TO4:A(Q)=0:D(Q)=0:B(Q)=0:V(Q)=0:C(Q)=0:NEXTQ:Z=0:F=0
140 '***SCREEN SET UP***
150 CLS:PRINT@0,"BETS = ";M;:PRINT@48,"CREDITS = ";C;:PRINT@73,"
PAYOUTS";
160 PRINT@128,"TWO PAIR =";P(0);:PRINT@192,"3 / KIND =";P(1);:PR
INT@256,"STRAIGHT =";P(2);:PRINT@320,"FLUSH    =";P(3);
170 PRINT@144,"FULL HOUSE =";P(4);:PRINT@208,"4 / KIND   =";P(5)
;:PRINT@272,"STR/FLUSH  =";P(6);:PRINT@336,"RYL/FLUSH  =";P(7);
180 IFZ=1RETURN
190 FORR=1TO5:F=165
200 ONRGOSUB450,460,470,480,490:GOSUB1560
210 NEXTR
220 '***CHOOSING CARDS***
230 S=1:FORQ=0TO4
240 A(Q)=RND(13)
250 D(Q)=RND(4)
260 B(Q)=A(Q)+D(Q)*20
270 Y=0
280 IFB(Q)=C(Y)GOTO240
290 IFY=QTHENY=Y+1:IFY=5GOTO320 ELSE280
300 IFB(Q)=B(Y)GOTO240
310 Y=Y+1:IFY=5THENGOTO320ELSE280
320 IFS=0GOTO350
330 IFS=1THENNEXTQ
340 GOTO370
350 GOTO680
360 '***PLACING BETS***
370 PRINT@163,"PRESS";
380 PRINT@233,"B TO PLACE BETS";
390 PRINT@297,"D TO DRAW";
```

```
400 A$=" ":A$=INKEY$:IFA$=""GOTO400
410 IFA$="B"ANDC>=1THENC=C-1:M=M+1:PRINT@7,M;:PRINT@58,C;:GOTO40
0ELSEIFC<=0ANDA$="B"THENPRINT@361,"YOU CAN'T BET ANY MORE.";:PRIN
T@425,"YOU'LL HAVE TO DRAW";:GOTO400
420 IFA$="D"ANDM=0THENPRINT@361,"YOU MUST PLACE A BET";:FORQ=1TO
1000:NEXT:PRINT@361,CHR$(30)ELSEIFA$="D"GOTO1650
430 GOTO400
440 '***STARTING POINTS FOR DRAWING CARDS***
450 A=451:T=514:B=525:Y=963:Z=515:H=583:RETURN
460 A=463:T=526:B=537:Y=975:Z=527:H=595:RETURN
470 A=475:T=538:B=549:Y=987:Z=539:H=607:RETURN
480 A=487:T=550:B=561:Y=999:Z=551:H=619:RETURN
490 A=499:T=562:B=573:Y=1011:Z=563:H=631:RETURN
500 '***HOLDING CARDS***
510 PRINT@233,CHR$(30);:PRINT@297,CHR$(30);:PRINT@361,CHR$(30);:
PRINT@425,CHR$(30);
520 PRINT@233,"1 - 5 TO HOLD CARDS";
530 PRINT@297,"C TO CANCEL HELD CARDS";
540 PRINT@361,"D TO DRAW";
550 PRINT@392,"1";:PRINT@404,"2";:PRINT@416,"3";:PRINT@428,"4";:
PRINT@440,"5";
560 A$="":A$=INKEY$:IFA$=""GOTO560
570 IFA$="1"THENPRINT@390,"HELD";:V(0)=1
580 IFA$="2"THENPRINT@402,"HELD";:V(1)=1
590 IFA$="3"THENPRINT@415,"HELD";:V(2)=1
600 IFA$="4"THENPRINT@427,"HELD";:V(3)=1
610 IFA$="5"THENPRINT@439,"HELD";:V(4)=1
620 IFA$="C"THENPRINT@384,CHR$(30);:FORQ=0TO4:V(Q)=0:NEXTQ:GOTO5
50
630 IFA$="D"THEN660
640 GOTO560
650 '***REPRINTING THE NEW CARDS***
660 S=0:FORQ=0TO4
670 IFV(Q)=0THENC(Q)=B(Q):GOTO240
680 IFV(Q)=0THENR=Q+1:ONRGOSUB450,460,470,480,490:F=165:GOSUB162
0
690 NEXTQ
700 FORQ=0TO4
710 IFV(Q)=0THENR=Q+1:ONRGOSUB450,460,470,480,490:F=128:GOSUB162
0:GOSUB1700
720 NEXTQ
730 '***PUTTING CARDS IN ORDER***
740 FORI=1TO4
750 FORJ=0TOI-1
760 IFA(J)>A(I)THENK=A(I):A(I)=A(J):A(J)=K
770 NEXTJ
780 NEXTI
790 '***CHECKING FOR WINS***
800 T=1:V=0
810 ONTGOSUB850,890,940,1010,1050
820 IFV=0THENT=T+1:IFT<>6GOTO810
830 FORQ=1TO1000:NEXTQ:GOTO1090
840 '***FOUR OF A KIND***
850 IFA(0)=A(1)ANDA(1)=A(2)ANDA(2)=A(3)THENV=1
860 IFA(1)=A(2)ANDA(2)=A(3)ANDA(3)=A(4)THENV=1
870 RETURN
880 '***THREE OF A KIND AND FULL HOUSE***
890 IFA(0)=A(1)ANDA(1)=A(2)THENV=2:IFA(3)=A(4)THENV=V+1
900 IFA(1)=A(2)ANDA(2)=A(3)THENV=2
910 IFA(2)=A(3)ANDA(3)=A(4)THENV=2:IFA(0)=A(1)THENV=V+1
920 RETURN
930 '***TWO PAIR***
940 P=0:IFA(0)=A(1)P=P+1
950 IFA(1)=A(2)P=P+1
960 IFA(2)=A(3)P=P+1
970 IFA(3)=A(4)P=P+1
980 IFP=2THENV=4
990 RETURN
1000 '***FLUSH***
1010 IFD(0)=D(1)ANDD(1)=D(2)ANDD(2)=D(3)ANDD(3)=D(4)THENV=5:GOSU
B1050
1020 IFV=6THENV=8
1030 RETURN
1040 '***STRAIGHT***
1050 IFA(0)+1=A(1)ANDA(1)+1=A(2)ANDA(2)+1=A(3)ANDA(3)+1=A(4)V=V+
6
1060 IFA(0)=1ANDA(1)+A(2)+A(3)+A(4)=46THENV=6
1070 RETURN
```

```
1080 '***PRINTING UP WINNINGS***
1090 IFV=0GOTO1520
1100 CLS
1110 IFV=1THENE=P(5)*M:PRINT@400,"FOUR OF A KIND. COLLECT";E;
1120 IFV=2THENE=P(1)*M:PRINT@400,"THREE OF A KIND. COLLECT";E;
1130 IFV=3THENE=P(4)*M:PRINT@400,"FULL HOUSE. COLLECT";E;
1140 IFV=4THENE=P(0)*M:PRINT@400,"TWO PAIR. COLLECT";E;
1150 IFV=5THENE=P(3)*M:PRINT@400,"FLUSH. COLLECT";E;
1160 IFV=6THENE=P(2)*M:PRINT@400,"STRAIGHT. COLLECT";E;
1170 IFV=8THENE=P(7)*M:PRINT@400,"ROYAL FLUSH. COLLECT";E;
1180 IFV=11THENE=P(6)*M:PRINT@400,"STRAIGHT FLUSH. COLLECT";E;
1190 '***COLLECT OR DOUBLE UP***
1200 FORT=0TO7:P(T)=P(T)*2*M:NEXTT
1210 PRINT@464,"DOUBLE UP - PRESS D";
1220 PRINT@528,"COLLECT WINNINGS - PRESS C";
1230 A$=INKEY$:IFA$=""GOTO1230
1240 IFA$="C"THENC=C+E:M=0:GOTO120
1250 IFA$<>"D"THENA$="":GOTO1230
1260 '***DOUBLE UP***
1270 Z=1:CLS:GOSUB160:PRINT@7,"DOUBLE PAYOUTS";
1280 GOSUB470:GOSUB1560
1290 A(2)=RND(13)
1300 D(2)=RND(4)
1310 PRINT@169,"PRESS B FOR BIG ";
1320 PRINT@233,"(LARGER THAN EIGHT)";
1330 PRINT@297,"PRESS S FOR SMALL";
1340 PRINT@361,"(SMALLER THAN EIGHT)";
1350 A$=INKEY$:IFA$=""GOTO1350
1360 IFA$="B"PRINT@809,"YOUR GUESS IS BIG";:GOTO1390
1370 IFA$="S"PRINT@809,"YOUR GUESS IS SMALL";:GOTO1400
1380 GOTO1350
1390 IFA$="B"ANDA(2)>8THEN1420
1400 IFA$="S"ANDA(2)<8THEN1420
1410 GOTO1460
1420 R=3:F=128:GOSUB470:GOSUB1620:GOSUB1700
1430 FORQ=1TO1000:NEXTQ
1440 CLS:E=E*2:PRINT@400,"CONgRATULATIONS - YOU GUESSED CORRECTL
Y";:PRINT@464,"COLLECT";E;"  DOUBLE UP AGAIN PRESS D";
1450 FORQ=0TO7:P(Q)=P(Q)*2:NEXTQ:GOTO1220
1460 R=3:F=128:GOSUB470:GOSUB1620:GOSUB1700
1470 FORT=1TO1000:NEXTT
1480 CLS:PRINT@400,"YOU GUESSED INCORRECTLY";:FORT=1TO500:NEXT
1490 IFC<=0THENCLS:PRINT@390,"SORRY - YOU'VE GOT NOTHING ELSE TO
 PLAY WITH....";:FORQ=1TO3000:NEXT:CLS:END
1500 FORQ=1TO2000:NEXT
1510 M=0:GOTO120
1520 IFC<=0THEN1490ELSECLS:PRINT@338,"SORRY, YOU WON NOTHING";:P
RINT@466,"PRESS ANY KEY TO CONTINUE";
1530 A$=INKEY$:IFA$=""GOTO1530
1540 GOTO1510
1550 '***DRAWING UP CARDS***
1560 PRINT@A,STRING$(10,176);
1570 PRINT@Y,STRING$(10,131);
1580 PRINT@T,CHR$(170);:T=T+64:IFT>960GOTO1600
1590 GOTO1580
1600 PRINT@B,CHR$(149);:B=B+64:IFB>960THENF=165:GOTO1620
1610 GOTO1600
1620 PRINT@Z,STRING$(10,F);
1630 Z=Z+64:IFZ>960RETURN
1640 GOTO1620
1650 FORR=1TO5
1660 ONRGOSUB450,460,470,480,490
1670 F=128:GOSUB1620:GOSUB1700
1680 NEXTR
1690 GOTO510
1700 OND(R-1)GOSUB1720,1760,1790,1840:GOSUB1900
1710 RETURN
1720 H=H+61:PRINT@H,CHR$(184)+CHR$(191)+CHR$(180)+CHR$(176)+CHR$
(190)+CHR$(189)+CHR$(144);:H=H+64
1730 PRINT@H,CHR$(130)+CHR$(175)+STRING$(3,191)+CHR$(135);:H=H+6
6
1740 PRINT@H,CHR$(139)+CHR$(159)+CHR$(129);
1750 RETURN
1760 PRINT@H,CHR$(160)+CHR$(180);:H=H+63:PRINT@H,CHR$(184)+STRIN
G$(2,191)+CHR$(189)+CHR$(144);:H=H+63
1770 PRINT@H,CHR$(174)+CHR$(191)+CHR$(143)+CHR$(159)+CHR$(175)+C
HR$(191)+CHR$(132);:H=H+67:PRINT@H,CHR$(133);
1780 RETURN
1790 PRINT@H,CHR$(184)+CHR$(188)+CHR$(144);:H=H+63
```

```
1800 PRINT@H,CHR$(162)+CHR$(143)+CHR$(159)+CHR$(167);:H=H+63
1810 PRINT@H,CHR$(168)+CHR$(191)+STRING$(2,189)+CHR$(191)+CHR$(1
89);:H=H+65
1820 PRINT@H,CHR$(139)+CHR$(129)+CHR$(149)+CHR$(139)+CHR$(129);
1830 RETURN
1840 PRINT@H,CHR$(184)+CHR$(144);:H=H+62
1850 PRINT@H,CHR$(160)+CHR$(190)+STRING$(2,191)+CHR$(180);:H=H+6
3
1860 PRINT@H,CHR$(136)+STRING$(5,191)+CHR$(157);:H=H+65
1870 PRINT@H,CHR$(130)+CHR$(175)+STRING$(2,191)+CHR$(135);:H=H+6
6
1880 PRINT@H,CHR$(139)+CHR$(129);
1890 RETURN
1900 B$(R-1)=A$(A(R-1))
1910 PRINT@Z-448,B$(R-1);
1920 P=LEN(B$(R-1))
1930 J=Y-54-P
1940 PRINT@J,B$(R-1);
1950 RETURN
1960 CLS:PRINT@20,"** P O K E R **"
1970 PRINT:PRINT
"    THIS GAME IS VERY SIMPLE TO PLAY. YOU START OF WITH TEN
 CREDITS AND YOU BET AS MANY AS YOU LIKE ON EACH HAND. THEN YOU
DRAW THE CARDS AND SELECT THE CARDS YOU WANT TO HOLD. YOU THEN  D
RAW THE CARDS AGAIN AND YOU ARE TOLD IF YOU WON."
1980 PRINT:PRINT
"    IF YOU WIN YOU HAVE THE CHOICE TO DOUBLE UP OR TO COLLECT
 YOUR MONEY. IF YOU DOUBLE UP, YOU WAGER YOUR WINNINGS ON
WHETHER A CARD IS BIG ( GREATER THAN EIGHT ) OR WHETHER IT IS   S
MALL ( LESS THAN EIGHT ).";
1990 PRINT"IF YOU GUESS CORRECTLY, YOU WIN TWICE THE AMOUNT YOU
ORIGINALLY WON. THE GAME CONTINUES UNTILL YOU RUNOUT OF CREDITS.
G O O D    L U C K ....."
2000 PRINT:INPUT"PRESS <NEW LINE> TO START PLAYING.";L
2010 CLS:RETURN
```

***** SHIFTL   m/1 16K   by F. Ellett *****

Recently we wished to introduce a Quadriplegic friend of ours to computer programming. As she is able to use an electric typewriter with the aid of a wooden stick with a rounded knob on its end, we assumed the keyboard would present no extra difficulties for her.

However, our friend quickly discovered that without being able to lock the shift key down, shifted key tokens, like quotation marks, were not available to her.

And so the following program was developed. Firstly, as a simple scan reading for the shift key and then changing the next key to its shifted value. Then as it was apparent that we needed a fairly hefty key delay, we added your very useful Keytone delay and finally we added single key words for the alphabet characters.

As the program was written for a System 80 (to out sound to cassette port two) Tandy users should change byte 7FD7 from Ø5 to Ø1 and byte 7FDE from Ø6 to Ø2 to avoid toggling their cassette motor relay.

System 80 owners should be aware that sound is disabled after doing a CLOAD or CSAVE and need to perform an OUT 254,255 to relatch cassette port 2. We have chosen to use the U key for this command.

Shifted numeral Ø returns a null value and can be used to cancel a shifted key scan should shift be pressed in error.

Two finger typists should be aware that holding shift down whilst pressing another key will return its normal function. The program as presented resides at 7E5Ø to 7FEF, the entry point is 7E5Ø. Memory should be protected at 32335.

The program can be considered as 3 blocks and relies on the fact that each time BASIC scans the keyboard, it jumps to an address in reserved RAM (4Ø16H) before the interpreter processes the pressed key. By placing a jump to block 1. at 4Ø16H, it is possible to modify the value of the pressed key before it is passed to the interpreter for processing.

BLOCK 1. Lines 500-650.
Block 1 simply looks to see if the shift key has been pressed. This is done by calling the keyboard switch matrix at 388ØH. (The A register will contain Ø1H if the SHIFT key has been pressed in that scan).

If the SHIFT key has been pressed the jump address at 4016H is altered, forcing a jump to block 2 in each subsequent scan.

BLOCK 2. Lines 660-910
Block 2 knows SHIFT has been pressed, so it waits to capture the next key pressed. This is done by calling 03E3H each scan. (The A register will return with the ASCII value of any key pressed during that scan).

As a check against a two handed typist pressing both SHIFT and a letter together, the contents of the A register are masked so that values less than 2CH (comma) and greater than 5AH (capital Z) are returned unaltered.

Once a key within the desired range has been detected, the address at 4016H is altered yet again to point to Block 3.

Before control is returned though, the ASCII value of the key just pressed is used to access an address table and point to the character, or group of characters, that is to replace the pressed key. This address is saved in a special address buffer. The contents of A are zeroed then control is returned to BASIC.

BLOCK 3. Lines 920-1050

Block 3 is the write block. It calls 03E3H and then loads the A register with the replacement character, pointed to by the address buffer. It then increments the address buffer to point to the next character and returns to BASIC.

This routine is repeated over and over until a byte is discovered which has bit 7 set. Bit 7 is not used for ASCII characters and so makes a convenient end of text marker.

Once detected, the character is restored to its correct value by resetting bit 7, and the address at 4016H is changed for the last time to point again to the initial scan routine in Block 1.

That completes the substitution phase and the computer will operate in a normal fashion until the next shift key is detected. Because using a stylus to press the keys tended to cause a lot of key repeats, we felt it necessary to slow the keyscan rate down and chose to use an adaptation of the Keyboard Bleeper by C.E. Kendall (Micro-80 Nov. 80) as this had the added advantage of giving an audible indication of which key was pressed.

```
***** SHIFT LOCK *****
A PROGRAM TO ENABLE A DISABLED PERSON TO ENTER SHIFTED LETTERS
(C) F. ELLETT PALM BEACH, QLD. 1/8/1981
TO USE: PRESS AND RELEASE THE SHIFT KEY
THE NEXT KEY PRESSED WILL ASSUME ITS SHIFTED VALUE
KEYBOARD THEN RETURNS TO ITS NORMAL MODE
SHIFTED ALPHABET LETTERS WILL GIVE FULL KEY WORDS AS BELOW:-
    A= ASC          B= CLS          C= CSAVE"A"      D= DATA
    E= ELSE         F= FOR I=1TO     G= GOTO         H= GOSUB
    I= INPUT        J= ","           K= IK$= INKEY$ IF IK$=""THEN
    L= LEN          M= MID$(         N= NEXT         O= POKE
    P= PEEK(        Q= CHR$(         R= RETURN       S= STRING$(
    T= THEN         U= OUT254,255    V= VAL(         W= STR$(
    X= RIGHT$(      Y= RND(          Z= LEFT$(       0= NULL
    READY
    >
```

```
3C00              00100      ORG     3C00H
3C00 2A           00110      DEFM    '***** SHIFT LOCK *****'
3C40              00120      ORG     3C40H
3C40 41           00130      DEFM    'A PROGRAM TO ENABLE A DISABLED PERSON TO EN
TER SHIFTED LETTERS'
3C80              00140      ORG     3C80H
3C80 28           00150      DEFM    '(C) F. ELLETT PALM BEACH, QLD. 1/8/1981'
3CC0              00160      ORG     3CC0H
3CC0 54           00170      DEFM    'TO USE: PRESS AND RELEASE THE SHIFT KEY'
3D00              00180      ORG     3D00H
3D00 54           00190      DEFM    'THE NEXT KEY PRESSED WILL ASSUME ITS SHIFTE
D VALUE'
3D40              00200      ORG     3D40H
3D40 4B           00210      DEFM    'KEYBOARD THEN RETURNS TO ITS NORMAL MODE'
3D80              00220      ORG     3D80H
3D80 53           00230      DEFM    'SHIFTED ALPHABET LETTERS WILL GIVE FULL KEY
WORDS AS BELOW:-'
3DC0              00240      ORG     3DC0H
3DC0 41           00250      DEFM    'A= ASC          B= CLS          C= CSAVE"A"
D= DATA'
```

```
3E00                00260          ORG     3E00H
3E00 45             00270          DEFM    'E= ELSE            F= FOR I=1TO    G= GOTO
   H= GOSUB'
3E40                00280          ORG     3E40H
3E40 49             00290          DEFM    'I= INPUT           J= ","         K= IK$= INKEY
$ IF IK$=""THEN'
3E80                00300          ORG     3E80H
3E80 4C             00310          DEFM    'L= LEN             M= MID$(       N= NEXT
   O= POKE'
3EC0                00320          ORG     3EC0H
3EC0 50             00330          DEFM    'P= PEEK(           Q= CHR$(       R= RETURN
   S= STRING$('
3F00                00340          ORG     3F00H
3F00 54             00350          DEFM    'T= THEN            U= OUT254,255  V= VAL(
   W= STR$('
3F40                00360          ORG     3F40H
3F40 58             00370          DEFM    'X= RIGHT$(         Y= RND(        Z= LEFT$(
   O= NULL                                '
4020                00380          ORG     4020H
4020 803F           00390          DEFW    3F80H
7E50                00400          ORG     7E50H
7E50 E5             00410 INIT     PUSH    HL
7E51 21657E         00420          LD      HL,START
7E54 221640         00430          LD      (4016H),HL
7E57 21077F         00440          LD      HL,WORDT           ;START WORD TABLE
7E5A 22057F         00450          LD      (STORE1),HL        ;SAVE ADDRESS
7E5D 3EFF           00460          LD      A,OFFH             ;SET SYSTEM 80
7E5F D3FE           00470          OUT     (254),A            ;SECOND CASSETTE / SOUND
7E61 E1             00480          POP     HL
7E62 C3CC06         00490          JP      06CCH
7E65 CDE303         00500 START    CALL    03E3H
7E68 F5             00510          PUSH    AF                 ;SAVE IT FOR A SEC
7E69 3A8038         00520          LD      A,(3880H)          ;WHILE CHECK IF SHIFT
7E6C FE00           00530          CP      00                 ;PRESSED
7E6E 2007           00540          JR      NZ,NEXTKY          ;JUMP TO IT IF IT IS
7E70 F1             00550          POP     AF                 ;NOT FOR US
7E71 FE00           00560          CP      00
7E73 C4C27F         00570          CALL    NZ,SOUND1          ;ORDINARY KEY PRESSED
7E76 C9             00580          RET                        ;SEND IT BACK
7E77 E5             00590 NEXTKY   PUSH    HL
7E78 21847E         00600          LD      HL,WAIT            ;SET WAIT LOOP
7E7B 221640         00610          LD      (4016H),HL
7E7E E1             00620          POP     HL
7E7F F1             00630          POP     AF
7E80 CDC87F         00640          CALL    SOUND2             ;SHIFT KEY PRESSED
7E83 C9             00650          RET
7E84 CDE303         00660 WAIT     CALL    03E3H              ;LOOK FOR OUR KEY
7E87 FE00           00670          CP      00
7E89 C8             00680          RET     Z
7E8A E5             00690          PUSH    HL
7E8B FE2C           00700          CP      2CH                ;IS IT LESS THAN ,
7E8D 3845           00710          JR      C,GOBACK                  ;GO BACK IF LESS
7E8F FE5A           00720          CP      5AH                ;IS IT GREATER THAN Z
7E91 3041           00730          JR      NC,GOBACK          ;THEN GO BACK
7E93 DDE5           00740          PUSH    IX
7E95 DD21AA7E       00750          LD      IX,TABLE-2CH       ;POINT IX TO TABLE
7E99 329E7E         00760          LD      (FINDAD+2),A       ;SET IX TO CORRECT BYTE
7E9C DD7E00         00770 FINDAD   LD      A,(IX+00)
7E9F 2A057F         00780          LD      HL,(STORE1)        ;START OF WORD TABLE
7EA2 85             00790          ADD     A,L                ;ADD TABLE DISPLACEMENT
7EA3 6F             00800          LD      L,A                ;TO WORD TABLE POINTER
7EA4 3001           00810          JR      NC,NOCAR
7EA6 24             00820          INC     H                  ;BUMP IT IF CARRY
7EA7 2B             00830 NOCAR    DEC     HL                 ;CORRECT POINTER
7EA8 22057F         00840          LD      (STORE1),HL        ;SAVE WORD ADDRESS
7EAB 21B97E         00850          LD      HL,WRITE           ;SET WRITE LOOP
7EAE 221640         00860          LD      (4016H),HL
7EB1 AF             00870          XOR     A                  ;NULL UNWANTED CHARACTER
7EB2 DDE1           00880          POP     IX
7EB4 E1             00890          POP     HL
7EB5 CDCE7F         00900          CALL    SOUND3             ;A SHIFTED KEY PRESSED
7EB8 C9             00910          RET
7EB9 E5             00920 WRITE    PUSH    HL
7EBA 2A057F         00930          LD      HL,(STORE1)        ;START WORD ADDRESS
7EBD 23             00940          INC     HL
7EBE 22057F         00950          LD      (STORE1),HL        ;POINT TO NEXT LETTER
7EC1 7E             00960          LD      A,(HL)             ;SEND HER OFF
7EC2 CB7F           00970          BIT     7,A                ;IS IT LAST CHARACTER?
```

```
7EC4 280E      00980          JR      Z,GOBACK        ;CARRY ON IF NOT
7EC6 CBBF      00990          RES     7,A             ;TOGGLE BYTE IF LAST ONE
7EC8 21077F    01000          LD      HL,WORDT        ;RESTORE ADDRESS
7ECB 22057F    01010          LD      (STORE1),HL
7ECE 21657E    01020          LD      HL,START        ;RESET KEYBOARD JUMP
7ED1 221640    01030          LD      (4016H),HL
7ED4 E1        01040 GOBACK   POP     HL
7ED5 C9        01050          RET
               01060 ;TABLE STARTS HERE
7ED6 00        01070 TABLE    DEFB    00
7ED7 01        01080          DEFB    WT1-WORDT
7ED8 02        01090          DEFB    WT2-WORDT
7ED9 03        01100          DEFB    WT3-WORDT
7EDA 0F        01110          DEFB    WTO-WORDT
7EDB 04        01120          DEFB    WT4-WORDT
7EDC 05        01130          DEFB    WT5-WORDT
7EDD 06        01140          DEFB    WT6-WORDT
7EDE 07        01150          DEFB    WT7-WORDT
7EDF 08        01160          DEFB    WT8-WORDT
7EE0 09        01170          DEFB    WT9-WORDT
7EE1 0A        01180          DEFB    WT10-WORDT
7EE2 0B        01190          DEFB    WT11-WORDT
7EE3 0C        01200          DEFB    WT12-WORDT
7EE4 0D        01210          DEFB    WT13-WORDT
7EE5 0E        01220          DEFB    WT14-WORDT
7EE6 00        01230          DEFB    00
7EE7 01        01240          DEFB    WT1-WORDT
7EE8 02        01250          DEFB    WT2-WORDT
7EE9 03        01260          DEFB    WT3-WORDT
7EEA 04        01270          DEFB    WT4-WORDT
7EEB 10        01280          DEFB    A1-WORDT
7EEC 14        01290          DEFB    B1-WORDT
7EED 18        01300          DEFB    C1-WORDT
7EEE 20        01310          DEFB    D1-WORDT
7EEF 24        01320          DEFB    E1-WORDT
7EF0 29        01330          DEFB    F1-WORDT
7EF1 36        01340          DEFB    G1-WORDT
7EF2 3B        01350          DEFB    H1-WORDT
7EF3 41        01360          DEFB    I1-WORDT
7EF4 47        01370          DEFB    J1-WORDT
7EF5 4A        01380          DEFB    K1-WORDT
7EF6 65        01390          DEFB    L1-WORDT
7EF7 69        01400          DEFB    M1-WORDT
7EF8 6E        01410          DEFB    N1-WORDT
7EF9 73        01420          DEFB    O1-WORDT
7EFA 78        01430          DEFB    P1-WORDT
7EFB 7D        01440          DEFB    Q1-WORDT
7EFC 82        01450          DEFB    R1-WORDT
7EFD 89        01460          DEFB    S1-WORDT
7EFE 91        01470          DEFB    T1-WORDT
7EFF 96        01480          DEFB    U1-WORDT
7F00 A1        01490          DEFB    V1-WORDT
7F01 A5        01500          DEFB    W1-WORDT
7F02 AA        01510          DEFB    X1-WORDT
7F03 B1        01520          DEFB    Y1-WORDT
7F04 B5        01530          DEFB    Z1-WORDT
7F05 0000      01540 STORE1   DEFW    0000
               01550 ;WORD TABLE STARTS HERE
7F07 BC        01560 WORDT    DEFB    3CH+80H  ;<
7F08 BD        01570 WT1      DEFB    3DH+80H  ;=
7F09 BE        01580 WT2      DEFB    3EH+80H  ;>
7F0A BF        01590 WT3      DEFB    3FH+80H  ;?
7F0B A1        01600 WT4      DEFB    21H+80H  ;!
7F0C A2        01610 WT5      DEFB    22H+80H  ;"
7F0D A3        01620 WT6      DEFB    23H+80H  ;#
7F0E A4        01630 WT7      DEFB    24H+80H  ;$
7F0F A5        01640 WT8      DEFB    25H+80H  ;%
7F10 A6        01650 WT9      DEFB    26H+80H  ;&
7F11 A7        01660 WT10     DEFB    27H+80H  ;'
7F12 A8        01670 WT11     DEFB    28H+80H  ;(
7F13 A9        01680 WT12     DEFB    29H+80H  ;)
7F14 AA        01690 WT13     DEFB    2AH+80H  ;*
7F15 AB        01700 WT14     DEFB    2BH+80H  ;+
7F16 80        01710 WTO      DEFB    80H          ;NULL CHARACTER
7F17 41        01720 A1       DEFM    'ASC'
7F1A A8        01730          DEFB    0A8H
```

```
7F1B 43          01740 B1      DEFM    'CLS'
7F1E A0          01750         DEFB    0A0H
7F1F 43          01760 C1      DEFM    'CSAVE"A'
7F26 A2          01770         DEFB    0A2H
7F27 44          01780 D1      DEFM    'DAT'
7F2A C1          01790         DEFB    0C1H
7F2B 45          01800 E1      DEFM    'ELSE'
7F2F A0          01810         DEFB    0A0H
7F30 46          01820 F1      DEFM    'FOR I = 1 TO'
7F3C A0          01830         DEFB    0A0H
7F3D 47          01840 G1      DEFM    'GOTO'
7F41 A0          01850         DEFB    0A0H
7F42 47          01860 H1      DEFM    'GOSUB'
7F47 A0          01870         DEFB    0A0H
7F48 49          01880 I1      DEFM    'INPUT'
7F4D A0          01890         DEFB    0A0H
7F4E 22          01900 J1      DEFM    '",'
7F50 A2          01910         DEFB    0A2H
7F51 49          01920 K1      DEFM    'IK$=INKEY$: IF IK$="" THEN'
7F6B A0          01930         DEFB    0A0H
7F6C 4C          01940 L1      DEFM    'LEN'
7F6F A8          01950         DEFB    0A8H
7F70 4D          01960 M1      DEFM    'MID$'
7F74 A8          01970         DEFB    0A8H
7F75 4E          01980 N1      DEFM    'NEXT'
7F79 A0          01990         DEFB    0A0H
7F7A 50          02000 O1      DEFM    'POKE'
7F7E A0          02010         DEFB    0A0H
7F7F 50          02020 P1      DEFM    'PEEK'
7F83 A8          02030         DEFB    0A8H
7F84 43          02040 Q1      DEFM    'CHR$'
7F88 A8          02050         DEFB    0A8H
7F89 52          02060 R1      DEFM    'RETURN'
7F8F A0          02070         DEFB    0A0H
7F90 53          02080 S1      DEFM    'STRING$'
7F97 A8          02090         DEFB    0A8H
7F98 54          02100 T1      DEFM    'THEN'
7F9C A0          02110         DEFB    0A0H
7F9D 4F          02120 U1      DEFM    'OUT254,255'
7FA7 A0          02130         DEFB    0A0H
7FA8 56          02140 V1      DEFM    'VAL'
7FAB A8          02150         DEFB    0A8H
7FAC 53          02160 W1      DEFM    'STR$'
7FB0 A8          02170         DEFB    0A8H
7FB1 52          02180 X1      DEFM    'RIGHT$'
7FB7 A8          02190         DEFB    0A8H
7FB8 52          02200 Y1      DEFM    'RND'
7FBB A8          02210         DEFB    0A8H
7FBC 4C          02220 Z1      DEFM    'LEFT$'
7FC1 A8          02230         DEFB    0A8H
7FC2 E5          02240 SOUND1  PUSH    HL
7FC3 216030      02250         LD      HL,3060H
7FC6 180A        02260         JR      SOUND4
7FC8 E5          02270 SOUND2  PUSH    HL
7FC9 218030      02280         LD      HL,3080H
7FCC 1804        02290         JR      SOUND4
7FCE E5          02300 SOUND3  PUSH    HL
7FCF 219030      02310         LD      HL,3090H
7FD2 F5          02320 SOUND4  PUSH    AF
7FD3 C5          02330         PUSH    BC
7FD4 4C          02340         LD      C,H
7FD5 45          02350 SOUND5  LD      B,L
7FD6 3E05        02360         LD      A,05
7FD8 D3FF        02370         OUT     (0FFH),A
7FDA 10FE        02380 LOOP1   DJNZ    LOOP1
7FDC 45          02390         LD      B,L
7FDD 3E06        02400         LD      A,06
7FDF D3FF        02410         OUT     (0FFH),A
7FE1 10FE        02420 LOOP2   DJNZ    LOOP2
7FE3 0D          02430         DEC     C
7FE4 20EF        02440         JR      NZ,SOUND5
7FE6 C1          02450         POP     BC
7FE7 F1          02460         POP     AF
7FE8 E1          02470         POP     HL
7FE9 C9          02480         RET
7E50            02490         END     INIT
00000 TOTAL ERRORS
```

***** MORSE CODE DECODER L2/4K   by R. Edwards *****

This program allows you to enter a series of dots and dashes representing morse code characters. It then translates those characters into the letters of the alphabet they represent.

The program works by converting an inputted string of dots, dashes and spaces into a decimal number by segments (each morse code letter) through a type of "binary" system.

Firstly a substring is taken between spaces, which is a morse code letter, the dots are equivalent to a Ø and dashes to a 1. Then this number is equated by raising 2 to this number plus its position from the right so that "." = 1, "-" = 2, ".." = 3 (left dot = 2, right dot = 1), "--" = 6, (left dash = 4, right dash = 2). This is more complicated than a simple binary conversion but was necessary to avoid disimilar letter being confused.

Line 170 finds each space.
Line 180 keeps track of the first and second space and if one follows another a space is left on the screen (new word).
Line 190 defines some variables and the substring C$ (the letter) and with line 200 steps through the string and does the abovementioned conversion.
Line 200 as above, then does a first check for the validity of input.
Line 210 is the second validity check if input is involved a branch to line 240 is made.
Line 220 prints the calculated character.
Line 230 checks for end of string, if so, awaits for a restart.
Line 240 is the branch when an error has been made, which requests a correction to be made.
Lines 250-320 is the input/correcting subroutine. If requested to correct input it will not allow you to alter any part of the string that has already been calculated. This subroutine may be used (with minor alterations) in other programs.

```
20 CLS:CLEAR1000:DIMA$(120):FORI=1TO28:READA$(I):NEXT:FORI=1TO18
:READX,A$(X):NEXT
30 DATAE,T,I,A,N,M,S,U,R,W,D,K,G,O,H,V,F,,L,,P,J,B,X,C,Y,Z,Q
40 DATA120,,31,5,32,4,34,3,38,2,46,1,47,6,48,=,49,/,55,7,59,8,61
,9,62,0,75,?,84,.,105,;,114,",",119,":"
50 PRINTTAB(15)"MORSE CODE DECODER  -   INSTRUCTIONS
"TAB(15)"-----------------------------------

THIS PROGRAM USES A SIMPLE LINE EDITOR FOR THE INPUT AND
CORRECTION OF THE CODED MESSAGE. A GRAPHIC BLOCK WILL FLASH"
60 PRINT"OVER THE SPACE OR CHARACTER, WHICH CAN BE CHANGED BY TY
PING
OVER IT. THE < AND > KEYS (,&.) CAN BE USED TO MOVE THE CURSOR,
SHIFT < WILL DELETE THE CURRENT CHARACTER, AND SHIFT > WILL
INSERT A SPACE. HIT ENTER TO RUN."
70 PRINT"
FOR THE INPUT OF MORSE CODE, TYPE THE K CHARACTER FOR DOTS AND
THE L CHARACTER FOR DASHES. INSERT A SPACE BETWEEN EACH LETTER,
AND TWO SPACES BETWEEN EACH WORD.":GOTO150
80 CLS:PS=-1:P2=1:P1=1:P=1:A$=" ":J=0:L=1:GOSUB170
90 J=J+1:IFMID$(A$,J,1)<>" "THEN90
100 P1=P2:P2=J+1:IFP1+1=P2THENC=120:GOTO140
110 C=0:L=P2-P1-1:C$=MID$(A$,P1,L):FORI=1TOL:X=1:IFMID$(C$,I,1)=
"."THENX=0
120 C=C+2[(X+L-I):NEXT:IFC<00RC>119GOTO160
130 IFA$(C)=""GOTO160
140 PS=PS+1:PRINT@PS,A$(C);:X=PS-INT(PS/64)*64:IFX>58PRINT:PS=PS
+64-X
150 IFJ<LEN(A$)THEN90ELSEPRINT@960,"PRESS ENTER TO CONTINUE";:IN
PUTQ$:GOTO80
160 PRINT@960,"ERROR - PLEASE CORRECT OR DELETE";:P=P1:L=LEN(A$)
:GOSUB170:PRINT@960,CHR$(31);:P2=P1:J=P1-1:GOTO90
170 C=255+P:PRINT@C,CHR$(143);:FORI=1TO25:S$=INKEY$:IFS$=""NEXT:
PRINT@C,MID$(A$,P,1);:FORI=1TO25:S$=INKEY$:IFS$=""NEXT:GOTO170
180 PRINT@C,MID$(A$,P,1);:Q=ASC(S$):IFQ=32ORQ=75ORQ=76THENQ=-1:G
OTO200ELSEIFQ=13RETURNELSEIFQ=44ANDP>P1THENP=P-1ELSEIFQ=46ANDP<LT
HENP=P+1ELSEIFQ=60ANDP<LTHENS$="":GOTO210ELSEIFQ=62THENS$=" "+MID
$(A$,P,1):GOTO210
190 GOTO170
200 IFS$="K"THENS$="."ELSEIFS$="L"THENS$=CHR$(95)
210 A$=LEFT$(A$,P-1)+S$+MID$(A$,P+1,L-P+1)
220 L=LEN(A$):PRINT@255+P,RIGHT$(A$,L-P+1)" ";:IFL>250THENA$=LEFT
$(A$,249)+" ":P=P-1:GOTO220
230 IFQ=-1THENP=P+1:IFP>LEN(A$)THENA$=A$+" "
240 GOTO170
```

***** NEXT MONTH'S ISSUE *****

Next month's issue will contain at least the following programs plus the usual features and articles.

** TIC TAC TOE LI/4K **

Now this old family favourite has been written to run in a Level 1 4K machine. Play against your level 1 machine, complete with a graphic playing board.

** ATTACK LII/16K **

Shoot down the invaders before they bomb you. A two player space game, complete with sound effects and real time action as you try to beat the invader and your opponent.

** MULTIPLE REGRESSION LII/4K **

This is the third program in the series of scientific programs. This one relates a dependant variable with two other independent variables in respect to a surface and if any of that means anything to you, then this program is just for you.

** TEXT TYPER LII/16K **

A simple word processor program written in BASIC that lets you type in text with or without justification, save your text to tape and load it back, edit your text and lots more.

** BASIC LINE VALIDATOR m/1 **

This machine language program resides in an unused RAM area and when called it checks through your program for any unreferenced line numbers, reporting any errors that are found.

** ATOMIC TABLES LII/16K **

Students studying the subject of Atomic tables and Atomic numbers or anybody that is just interested, will find this program very useful. Did you know that there was an element called TECHNETIUM? We didn't until we ran this program!

APPLICATION FOR PUBLICATION OF A PROGRAM IN MICRO-80

Date ....

Tick where appropriate

To **MICRO-80**
Please consider the enclosed program for...

(i)   Publication in MICRO-80

(ii) Publication on disk or cassette only

(iii) Both

Name ....

Address ....

Postcode ....

* * * CHECK LIST * * *

Please ensure that the cassette or disk is clearly marked with your name and address, program name(s), Memory size, Level I, II, System 1 or 2, Edtasm, System, etc. The use of REM statements with your name and address is suggested, in case the program becomes separated from the accompanying literature.

Ensure that you supply adequate instructions, notes on what the program does and how it does it, etc.

For system tapes, the start, end, and entry points, etc.

The changes or improvements that you think may improve it.

Please package securely — padabags are suggested — and enclose stamps or postage if you want your cassette or disk returned.

***** CASSETTE EDITION INDEX *****

The cassette edition of MICRO-80 contains all the software listed each month, on cassette. All cassette subscribers need do is CLOAD and RUN the programs. Level II programs are recorded on side 1 of the cassette. Level I programs are recorded on side 2. Level I programs are not compatible with the System 80. All programs are recorded twice in succession. Note, System 80 computers have had different tape-counters fitted at different times. The approximate start positions shown are correct for the very early System 80 without the volume control or level meter. They are probably incorrect for later machines. The rates for a cassette subscription are printed on the inside front cover of each issue of the magazine.

The disk edition contains all those programs which can be executed from disk, including Level I programs. Level I disk programs are saved in the NEWDOS format. Users require the Level I/CMD utility supplied with NEWDOS + or NEWDOS 80 version 1.0 to run them.

				APPROX. START POSITION		
SIDE ONE	TYPE	I.D.	DISK FILESPEC	CTR-41	CTR-80	SYSTEM-80
MORSE DECODER	LII/4K	M	MORSE/BAS	15	10	10
"	"	"	"	37	25	26
POKER	LII/16K	P	POKER/BAS	60	40	42
"	"	"	"	118	80	84
PUT COMMAND	EDTASM	PUT	PUT/EDT	173	117	123
"	"	"	"	188	127	134
"	SYSTEM	"	PUT/CMD	200	136	143
"	"	"	"	205	139	146
SHIFT LOCK	EDTASM	SHIFTL	SHIFTL/EDT	210	142	150
"	"	"	"	255	172	180
"	SYSTEM	"	SHIFTL/CMD	296	200	210
"	"	"	"	310	209	220
SIDE TWO						
SHORTEN	EDTASM	SHORTN	SHORTEN/EDT	15	10	10
"	"	"	"	38	26	27
"	SYSTEM	"	SHORTEN/CMD	62	42	44
"	"	"	"	70	47	50

MICRO-80

LEVEL II ROM
REFERENCE MANUAL

by Edwin Paay

Published by MICRO-80 PRODUCTS

Written by Eddy Paay, the LEVEL II ROM REFERENCE MANUAL is the most complete explanation of the Level II BASIC interpreter ever published.

Part 1 lists all the useful and usable ROM routines, describes their functions explains how to use them in your own machine language programs and notes the effect of each on the various Z 80 registers.

Part 1 also details the contents of system RAM and shows you how to intercept BASIC routines as they pass through system RAM. With this knowledge, you can add your own commands to BASIC, for instance, or position BASIC programs in high memory—the only restriction is your own imagination!

Part 2 gives detailed explanations of the processes used for arithmetical calculations, logical operations, data movements, etc. It also describes the various formats used for BASIC, SYSTEM and EDITOR/ASSEMBLER tapes. Each section is illustrated by sample programs which show you how you can use the ROM routines to speed up your machine language programs and reduce the amount of code you need to write.

The LEVEL II ROM REFERENCE MANUAL is intended to be used by machine language programmers. It assumes a basic understanding of the Z 80 instruction set and some experience of Assembly Language programming. But BASIC programmers too will benefit from reading it. They will gain a much better insight into the functioning of the interpreter which should help them to write faster, more concise BASIC programs.

MICRO-80